# МОНИТОР

журнал для программистов-практиков

4.93

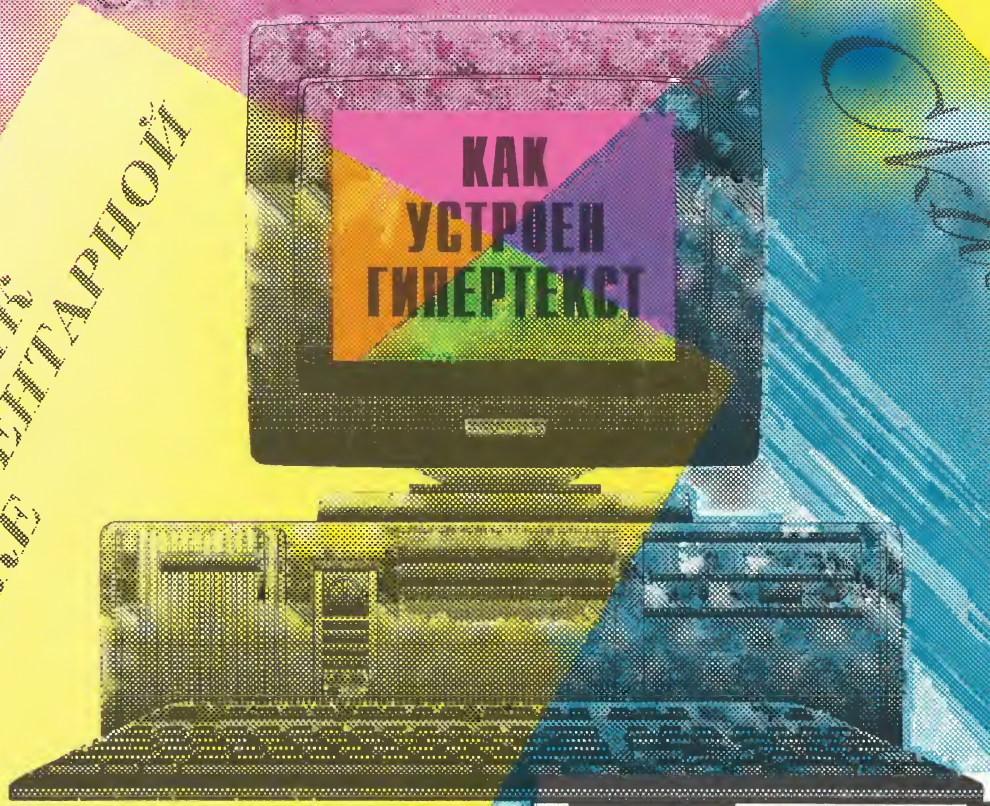
ISSN 0869-3889

ПУТЕВОДИТЕЛЬ ПО МОСКВЕ

ЭНЦИКЛОПЕДИЧЕСКИЙ СЛОВАРЬ

КАК УСТРОЕН ГИПЕРТЕКСТ

СПРАВОЧНИК ПО ЭЛЕМЕНТАРНОЙ ФИЗИКЕ



Сказки

LAW

Правила регистрации программы в США

GAMES

Подземный город

PROGRAMMER

QuickTools: инструменты для Paradox

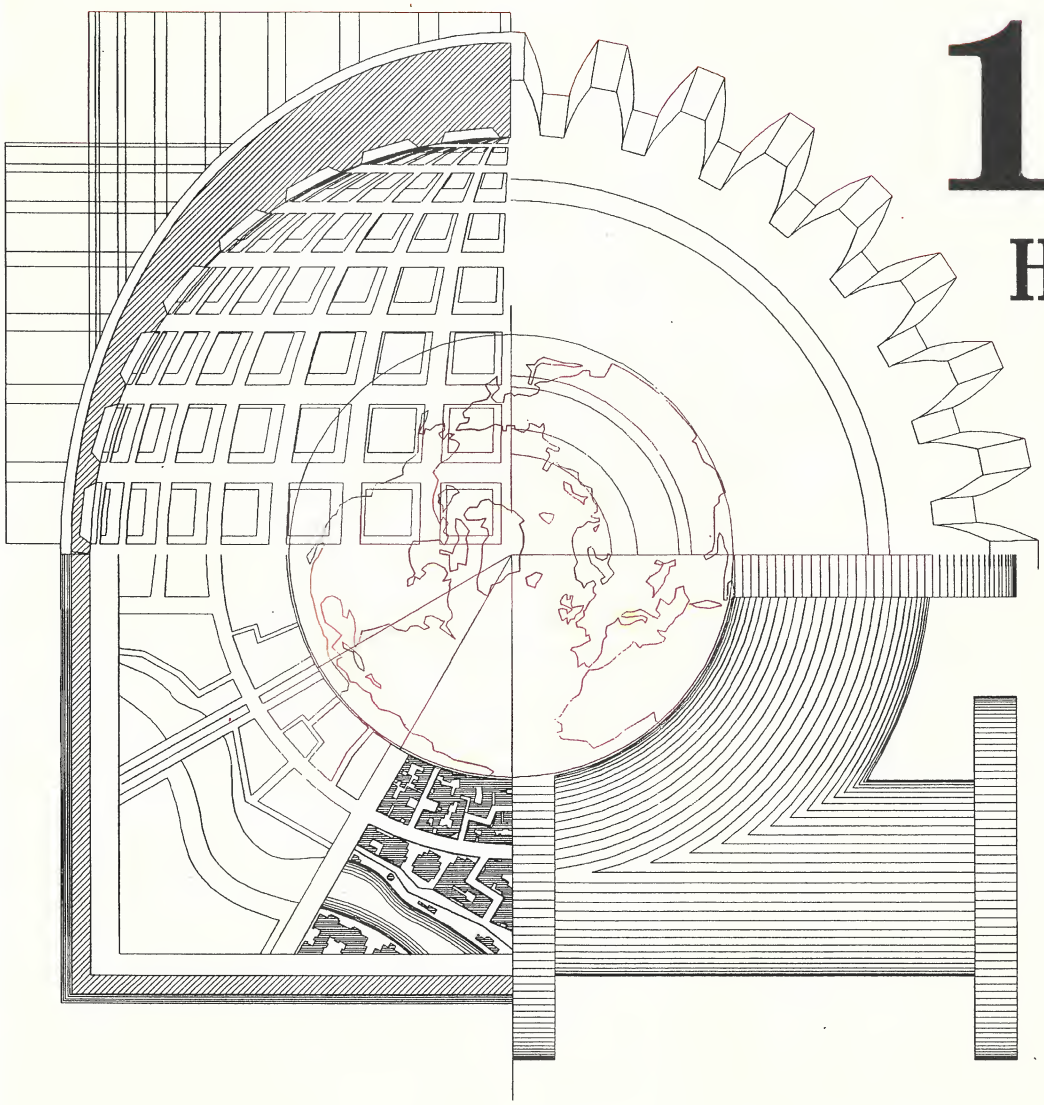
Создание демонстрационных программ

Стереографика

# АВТОКАД™

## ВЕРСИИ 12

## Ваши конкуренты уже купили Автокад 12!



# 174

## НОВИНКИ

Autodesk®

**Autodesk RF, Россия, 107005, Москва,**
**2-я Бауманская ул., д. 9/23, корп. 18**
**Телефон: 261 6363 Факс: 267 9927**

*Многие западные компьютерные авторитеты считают, что мы находимся на пороге новой технологической революции, которая случится на стыке между персональными компьютерами, телекоммуникациями и потребительской электроникой. Символом этой революции станет небольшое — размером с ладонь — устройство, которое по-английски называют «personal digital assistents» (PDA), «information appliances» или «personal communicators». Русского названия для них пока что не придумали. PDA являются результатом скрещивания компьютера, факса, сотового телефона и электронного блокнота.*

*Первый прототип PDA, названный Newton, уже был продемонстрирован фирмой Apple, однако он представляет собой скорее игрушку, показывающую потенциальные возможности новой технологии. Кроме Apple, разработку собственных версий PDA ведут Motorola, Sony, AT&T, Matsushita, Philips, Amstrad, EO и ряд других фирм. Первые промышленные образцы должны появиться уже в этом году. Предполагается, что к концу столетия PDA образуют новый колоссальный по объему рынок с ежегодными продажами в миллиарды долларов, сопоставимый по объему с рынком телефонных аппаратов или телевизоров.*

*Что такое PDA? Внешне — небольшая, помещающаяся на ладони коробочка с сенсорным экраном и электронным пером. Клавиатура отсутствует вследствие малых размеров устройства.*

*Функционально PDA, во-первых, электронный блокнот, способный настраиваться на почерк владельца, распознавать и записывать в память рукописные тексты. Таким образом он может служить личной базой данных, календарем, записной книжкой и полностью заменить собой бумажные средства планирования и организации личной работы. Помимо текстов, которые при вводе преобразуются в цифровую форму, PDA может хранить рисунки, запоминая их по точкам. Как тексты, так и рисунки вводятся с помощью электронного пера и сенсорного экрана.*

*Во-вторых, PDA включает необходимое для повседневной работы пользователя программное обеспечение, например, электронную таблицу или программу управления проектом. Управление работой программ осуществляется либо путем ввода рукописных команд, либо нажатием кончиком электронного пера на пиктограмму из меню на экране.*

*В-третьих, PDA содержит коммуникационные программы и электронные средства беспроводной связи наподобие сотовых телефонов, позволяющие сформировать и послать факсимильное сообщение в любую точку земного шара или подключиться через телефонную линию к любой доступной компьютерной сети, чтобы, например, оперативно узнать результаты торгов на бирже или заказать авиабилет.*

*Наконец, у PDA есть порт для стыковки с персональным компьютером и обмена с ним информацией и порт для подключения карт с дополнительной памятью или других устройств функционального расширения.*

*Пока что отсутствуют какие-либо стандарты на PDA и программное обеспечение для них, так что PDA разных фирм могут оказаться несовместимыми друг с другом и с существующими ПК. Однако 13 апреля 1993 года фирмы Compaq и Microsoft заключили соглашение о совместной разработке PDA и программного обеспечения для них, совместимого с операционной системой Microsoft Windows. Они обещают завершение проекта в срок от 6 до 18 месяцев.*

*Предположительные параметры первых PDA: размер — ладонь, вес — до 400 г, объем памяти — до 5000 страниц рукописного текста, цена — до 300 долларов.*

*В заключение — информация для наших подписчиков, подписавшихся на «Монитор» через агентство «Книга-Сервис». По взаимной договоренности между редакцией журнала «Монитор» и агентством «Книга-Сервис» редакция журнала «Монитор» приняла на себя все обязательства по рассылке журнала подписчикам, начиная с номера 2 за 1993 год. Поэтому мы просим Вас для разрешения всех вопросов, связанных с подпиской, обращаться непосредственно в редакцию «Монитора». Телефоны — в колонке справа.*


# МОНИТОР

**Журнал для программистов-практиков**

**Редколлегия:**

**Д.Э.Бройтман**
(главный редактор)

**О.А.Балдина**

**В.В.Водолазкий**

**В.С.Головин**

**Н.Н.Косарев**

**А.М.Миляков**
(ответственный секретарь)

**Н.А.Никитина**
(отдел рекламы)

**А.В.Орехов**
(издательский директор)

**Е.В.Равич**

**Л.И.Турок**

**Графическое бюро:**

**С.В.Макельский**
(главный художник)

**Г.Л.Карчикян**
(компьютерная верстка)

**Отдел распространения:**

**Р.А.Болдырев**

**А.А.Винокуров**

**Е.В.Гун**

**Представительство в Европе (Германия)**

**Марина Гетманская**
тел. 8-(0631)-7-49-05

**В оформлении номера использованы рисунки**

**А.С.Замулы**

Отпечатано
в ИПК "Московская правда"
Формат 60x84/8.
Тираж 30 000 экз.
Печать офсетная. Печ. л. 10,0
Цена договорная
Зак. 3192

**Издатель:**

**Научно-техническое общество**

Софт·Москва



Для писем: 117419 Москва, а/я 765
Тел. (095) 238-99-33
Факс (095) 238-32-78
Телекс 613322 GRANT SU
BBS (095) 135-14-39 ARSORBIS



© "Монитор", 1993

**Письма**

О том, как сгенерировать событие в момент старта программы в Turbo Vision, а также об одном способе формирования серийного номера программного продукта — 4

**Технологии программирования**

С. Курковский. Гипертексты с практической точки зрения. Часть 1 — 6

В статье рассказывается об основах организации гипертекстовых систем. Основное внимание уделено проблемам построения и применения гипертекстов. Предполагается опыт работы на C++. Первая часть посвящена вопросу создания гипертекстового транслятора.

А. Штрик. Отечественные инструментальные средства автоматизации создания прикладных программ — 16

Приведены интегральные характеристики отечественных инструментальных средств, представляющих разработки последних 5-7 лет. Рассмотрены конкретные отечественные инструментальные программные средства автоматизированной разработки прикладного программного обеспечения, представленные на проходивших в 1991-92 гг. выставках Софтул, Информатика и других.

С. Никоненко. Программируем СУБД на Turbo Vision Часть 2 — 22

Вторая часть цикла статей о библиотеке Data Vision. Обсуждаются принципы построения видимых элементов в рамках Turbo Vision. Приведен исходный текст программы-редактора файлов баз данных Dbase и Paradox.

**Базы данных**

С. Саркисян, Д. Шляйфер, Л. Ливеровский. Инструментарий QuickTools для создания прикладных систем в среде PARADOX — 32

В статье обсуждается несколько различных подходов к разработке инструментальных средств, используемых при создании так называемых заказных прикладных систем. Сформулированы основные требования, которым должны удовлетворять эти средства. Представлено краткое описание комплекса инструментальных средств QuickTools, обеспечивающего проектирование меню прикладных систем, создаваемых на базе СУБД Paradox 3.5.

**Компьютер в разрезе**

В. Хорт. О Clipper в стиле «ретро» — 38

Оригинальный подход к реализации поддержки «мыши» в Clipper программах, а также клавиш-модификаторов Ctrl, Alt и Shift.

**Аппаратура**

Д. Бройтман. **Микропроцессор Pentium Часть 2** — 42

Среда программирования Pentium с точки зрения программиста на ассемблере.

**DOS под микроскопом**

В. Плотников. **Создание демонстрационных программ в MS-DOS** — 46

В статье рассматриваются основные прерывания обслуживания клавиатуры в среде операционной системы MS-DOS. Излагаются принципы создания демонстрационных вариантов программ, использующих в процессе своего выполнения ввод с клавиатуры компьютера.

Д. Правиков. **Установка метки тома из программы** — 50

**Мир графики**

Д. Солдатенков. **Стереографика** — 51

Получение стереоскопических изображений на экране EGA монитора. Язык программирования: Turbo Pascal 6.0.

А. Аброськин. **Библиотека графических курсоров «мыши»** — 54

Описан набор графических курсоров манипулятора «мышь». Приведен исходный текст процедуры на языке ассемблера.

И. Завалей, И. Михайлов. **Выбери цвета...** — 56

В статье предложен способ управления цветовой палитрой, использующий механизм изменения интенсивности основных цветов RGB палитры.

**Право**

М. Корсаков. **Как зарегистрировать программу в США** — 60

В данной статье подробно рассматриваются процедурные вопросы регистрации компьютерных программ в США.

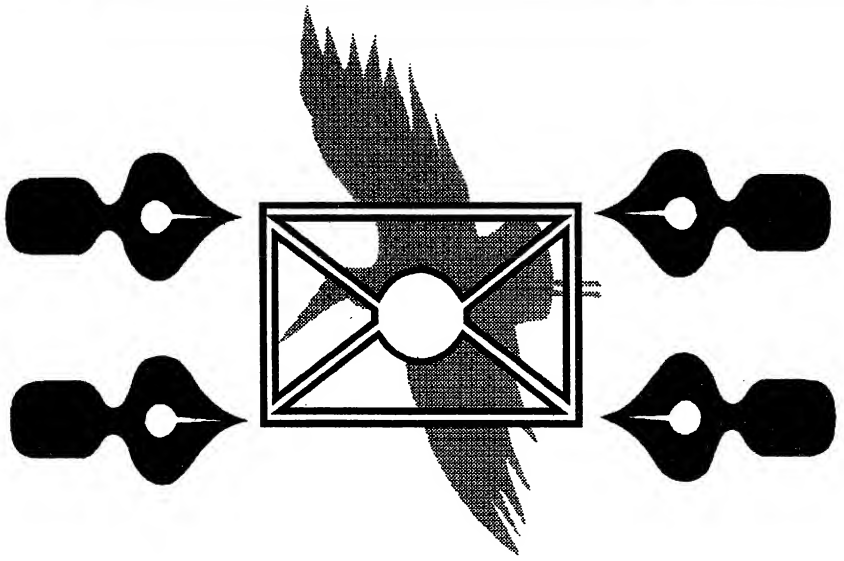
**Игры**

**Подземный город (по мотивам программы The Eye of Beholder)** — 64

**Бехолдер — порождение мрака (из досье фирмы TSR, Inc.)** — 70

**Этюды для программистов**

В. Очков **Двенадцать программ с дублями и эпиграфами, или три триады программирования, а точнее третий лишний, а второй неправильный** — 72



Уважаемый господин Водолазкий,
с удовольствием и пользой прочел Вашу статью о начальной установке обработчика событий Turbo Vision в «Мониторе» 4.92. Предлагаемый Вами путь действительно позволяет создавать цепочки «своих» событий, сколь угодно длинные и не обязательно в момент старта программы. Однако для генерации одного-единственного события в момент старта программы я просто перекрываю TApplication.Run:

```
procedure MyApp.Run;
begin
   Message(Application, evCommand,
                        cmAbout, nil);
   TApplication.Run;
end
```

Согласитесь, это несколько лаконичнее.

**С уважением**
**к.т.н. Анненков А.П.** ■

---

## Об одном способе формирования серийного номера программного продукта

Как известно, существует три основных подхода к защите программного обеспечения от несанкционированного использования.

Первый способ состоит в том, что держатели прав (copyright owners) на математическое обеспечение (МО) техническими способами не защищают его от несанкционированного использования, а целиком полагаются на юридические процедуры защиты своих прав.

Те, кто применяет второй способ, не рассчитывают на высокую эффективность юридических процедур защиты своих прав и различными техническими приемами препятствуют нелегальному тиражированию математического обеспечения (ключевые дискеты, которые не копируются стандартными средствами; привязка МО к уникальным характеристикам вычислительной установки, на которую оно установлено; аппаратные средства и так далее.)

Этот способ усложняет нелегальное использование МО (хотя и не исключает такового), но во многих случаях создает значительные дополнительные неудобства как для пользователей, так и для тех, кто сопровождает такой продукт.

Пользующиеся же третьим способом физически не препятствуют копированию МО, но каждую легально приобретенную копию программы снабжают уникальным номером, который при установке продукта необходимо указать.

Программа установки проверяет такой номер на допустимость (обычно он в закодированном виде хранится на дистрибутивном носителе). Затем, как правило, пользователь регистрируется у распространителя программы под этим номером.

Далее такой номер может выполнять несколько функций.

- Служить идентификатором нарушителя лицензионного соглашения (отвечать на вопрос: «Откуда ноги растут ?»).
- Использоваться для оценки эффективности работы распространителей МО, анализа спроса по регионам и так далее, если в структуре номера предусмотрено кодирование соответствующей информации, и решать какие-нибудь дополнительные задачи.

Чтобы такой номер эффективно использовался, он должен обладать высокой степенью защиты.

Это значит, что сконструировать допустимый номер (в случае, если правильный номер не хранится где-то на дистрибутивном носителе) или

исправить его в дистрибутиве (в противоположном случае) должно быть крайне затруднительно.

Каждый из способов имеет свои достоинства и недостатки. Выбор того или иного способа определяется следующими основными факторами:

- прогнозируемым соотношением между количеством легальных и нелегальных копий, находящихся в использовании, что, в свою очередь, определяется ценой продукта, его способностью решать проблемы пользователей и многими другими факторами;
- ценой самой защиты и степенью вносимых ею неудобств в работу с продуктом и в его сопровождение;
- эффективностью юридических процедур защиты авторских прав.

Например, известный продукт PARADOX (Borland International, Inc.), кроме функции регистрации легальной копии, использует номер для определения максимально допустимого количества пользователей, работающих в сети с программой. Сам по себе номер на дистрибутиве не хранится и только проверяется на допустимость программой установки PARADOX и программой обновления количества пользователей в сети NUPDATE.EXE.

Отдельный продукт LanPack, добавляющий сведения о пользователе к сетевой версии PARADOX, по сути, ничего не делает, а лишь проверяет номер на допустимость, извлекает из него информацию о добавляемом количестве пользователей к сетевому PARADOX и исправляет файл PARADOX.SOM, где эта информация хранится. Сама же сетевая версия PARADOX ничем не отличается от однопользовательской.

При активизации очередного пользователя сетевой версии информация об этом регистрируется в файле PARADOX.NET управления сетевым PARADOX и проверяется соответствие числа активных пользователей количеству, указанному в номере.

Однако номер PARADOX обладает очень малой степенью защиты. Достаточно любому программисту увидеть два номера, идущих подряд, и он, немного подумав, изобретет метод построения номера для любого количества пользователей в сети.

Приведем метод построения номеров целиком.

Для PARADOX версии 3.5 примерная структура номера такова:

```
xxVVVynsssssss , где
xxVVVy — префикс, причем
   VVV= 233 —  код версии PARADOX 3.0;
   VVV= 234 —  PARADOX 3.5;
   VVV= 235 —  PARADOX 4.0;
n — количество пользователей в сети:
   1–9 —  от одного до девяти пользователей
          (с шагом 1),
   A–I —  от 10 до 90 пользоватлей
          (с шагом 10).
sssssss — базовый номер.
```

Префикс строится произвольным образом (xx — любые буквы и цифры, y — буква от A до I). Затем подбирается базовый номер простым перебором вариантов от 1 до 10. Любой другой допустимый номер получается прибавлением числа 11*N (N=1,2,...) к базовому.

Например, для PARADOX 3.5:

AA234A10000009 — номер для одного пользователя (и далее номера 20, 31, ...);
ZZ234Z10000005 — номер для одного пользователя (и далее номера 16,27, ...);
AA234AI0000001 — номер для 90 пользователей;
ZZ234ZI0000008 — номер для 90 пользователей;
11234AI0000001 — номер для 90 пользователей.

Для PARADOX версии 3.0 и 4.0 метод построения номера такой же, меняется только код версии:

Например:

TC235HI0000003 — номер для 90 пользователей PARADOX 4.0.

Про номера, построенные подобным образом, нельзя сказать, могут ли (или могли) они быть присвоены какой-либо легальной копии PARADOX. Однако, использовав сконструированный таким образом номер для нелегальной копии продукта, уже нельзя вычислить, откуда произошла его «утечка».

Из сказанного можно сделать вывод, что используемая уважаемой фирмой схема не годится для того, чтобы техническими средствами заставлять приобретать PARADOX LanPack и выявлять нарушителей лицензионного соглашения.

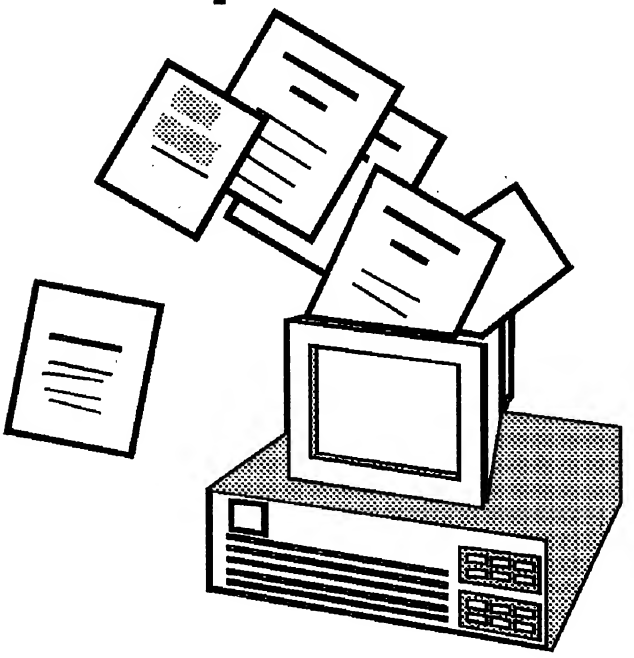
Видимо, разработчики не придавали схеме защиты большого значения, и фирма Borland International полагается на юридические процедуры защиты своих прав в случаях, когда их нарушения приносят серьезные убытки. При тиражах продукта, исчисляющихся сотнями тысяч экземпляров, такая политика, очевидно, является оправданной.

У некоторых отечественных пользователей, вероятно, появится вопрос: «А зачем же теперь вообще нужен LanPack?». Хотелось бы, чтобы такой вопрос ни у кого не возник. Но если все же он у кого-то появился, то позвольте вместо ответа напомнить банальную истину, что красть нехорошо.

Л. Андреев ■

# Гипертексты с практической точки зрения

*С. Курковский*



## Часть 1

Продолжая тему, начатую в работах «Технология создания оконного интерфейса» и «Меню как основа взаимодействия с пользователем», можно задать вопрос, а что же еще отличает профессионально написанную программу от непрофессиональной и делает программный продукт коммерчески успешным. Один из основных принципов, которым должен руководствоваться программист, ставя перед собой цель создать хорошую программу — принцип полной информированности пользователя о работе программы. Это подразумевает не только наличие подробной документации, хотя хорошая документация также является очень важным условием. Имеется в виду, что программный интерфейс должен обладать такими свойствами, чтобы информировать пользователя о текущем режиме работы, выполняемой задаче (а если такая задача решается долго, нужно каким-то образом дать понять, что процесс не зациклился). Кроме того, очень полезно иметь в программе возможность давать пользователю некоторую дополнительную информацию, а проще говоря — помощь. В больших многофункциональных программах пользователю просто необходимо получить помощь. При этом направляемая пользователю информация должна зависеть от конкретной ситуации, то есть помощь должна быть контекстно зависимой. Это, безусловно, несколько усложняет поставленную перед программистом задачу, но и результат использования контекстно зависимых систем помощи намного более впечатляющий.

Итак, подавляющее большинство профессионально написанных программных продуктов использует контекстно зависимые системы информации. В их число входят и все продукты фирмы Borland International, и всеми любимый Norton Commander, и многое другое. Но контекстно-зависимые системы информации применяются не только как системы помощи, но и как отдельные независимые программные продукты. Например, Norton Guides, для которого написано большое количество баз данных, и Tech Help! (by Flambeaux Software, Inc.) — прекрасный справочник по DOS и BIOS.

### Что такое гипертекст

Надеюсь, что после всего сказанного у читателя появилось желание узнать, а как же работает и как устроено то, что называется контекстно-зависимыми системами информации, или гипертекстами. **Гипертекстом** принято называть некоторый объем текстовой (иногда и не только текстовой) информации, состоящей из непересекающихся частей (мы будем называть их темами), между которыми установлены перекрестные ссылки (рис. 1). Безусловно, можно дать много других определений термина «гипертекст», что было прекрасно продемонстрировано в работе [1]. Поскольку настоящая статья посвя-

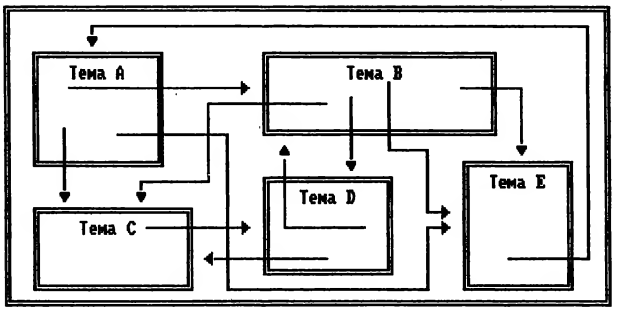



**Рис. 1. Общая структура гипертекста**

щена практическим проблемам и программированию систем поддержки гипертекстов, то данное здесь определение гипертекста может показаться несколько специфичным. Итак, **тема**, как любой текст, состоит из последовательности строк. В любом месте любой строки может находиться **гипертекстовая ссылка** на другую тему, таким образом, одна тема может иметь внутри себя неограниченное количество ссылок, и наоборот, количество ссылок на данную тему не ограничено. Каждая тема имеет уникальное имя — **заголовок темы**, кроме того, в любой момент можно получить список заголовков всех тем. Таким образом, гипертекст можно сравнить со справочником или энциклопедией, где темам соответствуют статьи, гипертекстовым ссылкам — ссылки на другие статьи, а списку заголовков всех тем соответствует оглавление. Этим можно ограничить минимальную характеристику структуры гипертекста. Но на практике очень часто возникает необходимость расширить, дополнить его возможности.

**Дополнение 1.** Как правило, гипертекст должен включать в себя одну или несколько специальных тем, которые соответствуют либо сообщению об ошибке, либо являются справочником по гипертексту (help on help — некоторая вводная статья в справочнике). Также можно включить в гипертекст некоторую строку, которая в дальнейшем будет иметь какое-то глобальное значение, например, эта строка может быть заголовком всего гипертекста.

**Дополнение 2.** Основа гипертекста (исходный текст), как и любой документ, подготавливается в текстовом редакторе и должна храниться в ASCII формате (хотя и не обязательно). Но что делать, если возникла крайняя необходимость поместить в текст какие-нибудь символы, которые нельзя напрямую ввести с помощью текстового редактора? Следовательно, необходимо обеспечить возможность закодировать такой символ с использованием других символов.

**Дополнение 3.** Гипертекст сам по себе несет только информацию о перекрестных ссылках (не считая собственно содержания текста). Почему бы не добавить туда еще какую-нибудь полезную информацию? Например, должным образом расставив специальные метки в тексте, можно определить смену текстовых атрибутов (цветов) или смену шрифтов для отображаемого текста.

**Дополнение 4.** Как при написании любого другого исходного текста, всегда полезно иметь возможность пользоваться комментариями. Это тоже надо учесть.

Безусловно, этот список неполон, и любой программист, творчески относящийся к своему труду, сможет внести добавления, адаптируя к собственным нуждам. В статье мы будем придерживаться данного определения и четырех дополнений, все программные разработки, приводимые здесь, также будут следовать определению.

## Формат исходного текста

Мы разработали структуру и характеристики гипертекста. Первый шаг в его практической реализации — создание формата исходного текста. Под исходным мы понимаем текст, в котором должным образом определены темы и их заголовки, гипертекстовые (перекрестные) ссылки, могут присутствовать определения специальных тем, специальных символов и атрибутов, могут стоять комментарии. От формата исходного текста зависят во многом и характеристики самого гипертекста, как качественные (простота и удобство в использовании), так и технологические (удобство машинной обработки, скорость доступа). Не будем изобретать велосипед, обратимся к уже имеющимся разработкам в этой области. Наиболее широко известным, распространенным и легким в использовании программным продуктом подобного рода является Norton Guides. Возможно, кто-нибудь из читателей имеет опыт написания собственных баз данных для Norton Guides.

Будем считать, что тема начинается с определения ее заголовка и продолжается до определения следующего заголовка или до конца исходного текста. На заголовок темы и гипертекстовую ссылку мы не будем накладывать никаких существенных ограничений, за исключением, естественно, длины строк и их количества. Для определения заголовков тем или ссылок можно использовать префиксы, то есть необходимо перед соответствующим словом в тексте поставить определенный символ. Возьмем символ «!» для имени темы, а для ссылки символ «&». В целях упрощения договоримся также, что определение заголовка темы может начинаться только с первой позиции в строке, а конец определения заголовка — это конец строки. Другими словами, заголовком темы является строка, начинающаяся с соответствующего префикса. Ссылка, в свою очередь, будет определяться как часть строки между двумя префиксами.

В примере 1 определяются две темы (с заголовками «тема 1» и «тема 2»), и во второй теме делается ссылка на первую.

**Пример 1. Определение тем и ссылок**

Исходный текст

```
|Тема 1
  Эта тема имеет номер 1, потому что она самая первая
|Тема 2
  Эта тема имеет номер 2, потому что перед ней
    стоит &тема 1&
```

Для задания атрибутов и специальных символов воспользуемся опытом Norton Guides, где для этого используются следующие комбинации:

а) ^Axx — для определения атрибута,

б) ^Cxx — для определения символа, где xx — шестнадцатиричное представление атрибута или ASCII кода символа.

Кроме того, в Norton Guides определялись и некоторые предопределенные атрибуты с помощью комбинаций ^b, ^u, ^r. Предопределенными они называются потому, что точные значения этих атрибутов определены в программе, и пользователь, составляя исходный текст, в явном виде их не указывает. В нашем гипертексте заданы пять таких атрибутов комбинацией символов ^1, ..., ^5. Конкретные значения этих атрибутов мы определим в исходном тексте. Для определения комментария можно воспользоваться символом «;», стоящим в первой позиции соответствующей строки. В примере 2 показано использование комментариев, атрибутов и символов.

**Пример 2. Использование комментариев, атрибутов и символов**

Исходный текст

```
^A1E Этот текст будет ЖЕЛТЫМ на СИНЕМ фоне
  Здесь я хочу изобразить музыкальные ноты ^C0D, ^C0E.
; Это очень глубокомысленный комментарий
```

Теперь, разработав формат исходного текста, мы вплотную подходим к вопросу о том, каким именно образом его лучше использовать. Есть две основные возможности:

а) напрямую интерпретировать исходный текст, не делая с ним никаких предварительных преобразований;

б) транслировать исходный текст в более удобный для программного восприятия формат (гипертекстовую базу данных) и лишь затем читать текст в этом формате.

Автору представляется наиболее предпочтительным второй подход.

## Гипертекстовый транслятор

**Гипертекстовым транслятором** мы назовем программу, имеющую на входе исходный текстовый файл, а на выходе — гипертекстовую базу данных. Анализируя исходный текст, транслятор должен уметь делать следующее:

1) находить все определения (тем, ссылок, специальных символов, атрибутов);

2) игнорировать строки, помеченные как комментарии;

3) проверять корректность всех определений.

В приведенной программе (листинг 1) главный цикл гипертекстового транслятора занимает всего 60 строк на C++. Но не все так просто, как кажется, хотя и не стоит лишний раз запугивать читателя трудностями программирования, лучше как следует все объяснить.

Для ввода данных предназначен объект класса HTInput (здесь и далее под аббревиатурой HT понимается HyperText). Функция ReadString просто читает строку текста из файла. С помощью функций Check... осуществляются проверки появления в тексте определений заголовков тем и ссылок (часть требования 1), комментариев (требование 2) и заголовка гипертекстовой сети (дополнение 1), который, кстати, мы определим как текстовую строку, начинающуюся с префикса «_ _TITLE:». Большинство функций класса HTInput тривиально и поэтому не заслуживает особого внимания.

Для решения всех четырех задач поступим следующим образом. В процессе работы транслятор будет создавать образ гипертекстовой сети, вид которой показан на рис. 2. По вертикали расположен список всех

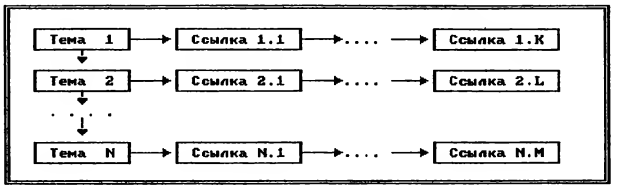



**Рис. 2. Структура для проверки корректности исходного текста (образ гипертекстовой сети)**

тем, определенных в тексте, по горизонтали — списки ссылок, сделанных в данной теме. Всю работу с сетью сосредоточим в классе HTNet, который занимается анализом и построением гипертекстовой сети, составленной из объектов класса HTNode (листинг 4). На описании класса HTNode остановимся несколько подробнее, так как именно этот класс является «самым маленьким кирпичиком» в программе. Поскольку данный класс используется также и в гипертекстовом интерпретаторе, сейчас нас будут интересовать не все поля объекта, а лишь основные, используемые в трансляторе (рис. 3).

Как следует из названия поля char *KeyWord, оно используется для хранения текстового имени темы/ссылки, полученного с помощью функции HTInput::ShortenWord. В случае если объект описывает тему, поля будут содержать следующую информацию:

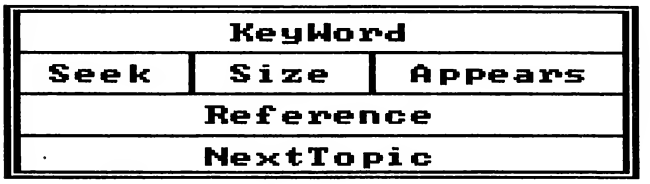



**Рис. 3. Структура класса HTNode (поля, используемые в гипертекстовом трансляторе)**

long Seek — адрес во временном файле (смещение от начала файла до начала текста данной темы);

int Size — размер текста темы в строках;

int Appears — номер строки исходного текста, где появляется заголовок темы/ссылки;

HTNode *NextTopic — указатель на следующую тему в сети (списке тем);

HTNode *Reference — указатель на первую ссылку из данной темы.

Если же в объекте описывается ссылка, то поля Seek, Size и NextTopic будут обнулены. Кроме конструктора и деструктора, в классе определены несколько операторов для сравнения объектов между собой и с текстовыми строками. Эти операторы используются только в гипертекстовом интерпретаторе.

Транслятор содержит один объект класса HTNet (листинг 5), целиком описывающий гипертекстовую сеть. Функция NewTopic конструирует новую тему, а функция NewReference — новые ссылки, сделанные из последней темы. Кроме построения сети, этот класс также занимается и ее анализом. Самые первые ошибки могут быть обнаружены еще на этапе построения сети. Например, при добавлении новой темы (функция NewTopic) делается проверка, не содержит ли список тем еще одну тему с таким же заголовком (функция InTopicList). Другая некорректная ситуация — перед текстом отсутствует заголовок темы (то есть этот текст не принадлежит ни к одной теме) — обнаруживается очень просто: при увеличении поля Size с помощью функции IncreaseTopicSize (что делается каждый раз при прочтении новой строки) будет обнаружено, что сеть пуста.

Предположим, что при построении сети ошибок не было, но это еще не означает, что все определения тем и ссылок полностью корректны. Очевидно, для того чтобы проверить корректность всех определений, надо зафиксировать все некорректные ситуации. В нашем случае такими будут следующие ситуации:

1) в тексте не определена ни одна тема;

2) в тексте определены две темы с одинаковыми заголовками;

3) в теме имеется ссылка на эту же тему (ссылка на себя);

4) определена ссылка на не определенную нигде тему.

Функции AnalyseNet и CheckReference занимаются выявлением этих некорректных ситуаций (номера соответствуют описанным).

1) Если не определена ни одна тема, вертикальный список будет пуст.

2) Для каждого элемента вертикального списка надо сделать проверку всех элементов, лежащих ниже, на совпадение заголовка темы.

3) Для каждого элемента горизонтального списка необходимо сделать проверку на совпадение с головным узлом (самым левым).

4) Для каждого горизонтального списка нужно сделать проверку по каждому элементу на совпадение с элементами вертикального списка.

Отсутствие совпадающего элемента из вертикального списка свидетельствует об ошибке. Отсутствие ошибок на этом этапе свидетельствует о полной корректности гипертекстовой сети.

## Как организовать базу данных

Проанализировав исходный текст и не найдя в нем ошибок, транслятор должен создать **гипертекстовую базу данных**. Хороший формат базы данных — тоже немаловажный фактор. От него зависит и быстрота доступа к различным полям (темам), и то, насколько просто осуществляется этот доступ. Для обеспечения максимального быстродействия можно создать специальный индексный файл с заголовками соответствующих тем, их размерами (в байтах или строках — кому как нравится) и адресами начала темы в базе данных. Можно и не создавать специальный файл, а поместить таблицу индексов в начало файла гипертекстовой базы данных (рис. 4). Именно таким образом построен класс HTOutput (листинг 6) — поддержка вывода гипертекстового транслятора, функция CreateDataBase запишет в базу данных заголовок — таблицу индексов. Теперь интерпретатору достаточно будет только один раз прочесть и запомнить эту таблицу, чтобы в дальнейшем использовать известные адреса для быстрого доступа к тексту. Подобный принцип организации базы данных даже для очень больших текстов (до 1 мегабайта) не будет снижать

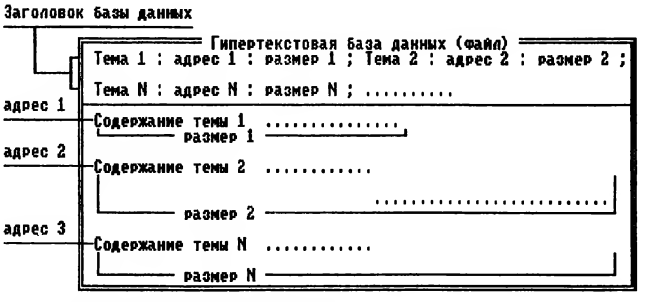



**Рис. 4. Организация базы данных**

быстродействия. Для проверки корректности (то есть того, что по этому адресу действительно находится начало нужного текста) можно в начале каждой темы ставить соответствующую метку (SYNCH_BYTE — байт синхронизации).

Еще несколько слов о работе функций класса HTOutput. Функция CreateDataBase, кроме таблицы индексов, записывает в заголовок метку базы данных (DBFILE_ID), количество тем в базе данных (TotalTopics), название гипертекстовой сети (NetTitle). Кроме того, каждый раз, когда транслятор обработал очередную строку исходного текста, эта строка записывается во временный файл, также поддерживаемый классом HTOutput (функция CompressLine). Гипертекст в базе данных можно хранить «один к одному», но лучше хранить его в сжатом виде. Одним из самых простых, но вместе с тем очень быстрых алгоритмов является алгоритм Run Length Encoding (RLE) — объединение последовательности одинаковых символов (байтов). Этот алгоритм дает хорошие результаты только в том случае, если в тексте используются отступы (не табуляция) и псевдографика (подчеркивания, таблицы, рамки). По этому алгоритму последовательность заменяется на метку кодирования, счетчик и собственно символ, который нужно повторить. Кроме применения алгоритма RLE, функция CompressLine также обрабатывает ссылки, спецсимволы и атрибуты, расставляя соответствующие метки. Создав заголовок, функция CreateDataBase копирует к нему временный файл, и все, база данных готова. Полная реализация гипертекстового транслятора приведена в листинге 7 (функция MakeHyperText).

В следующем номере журнала будет рассказано об основных принципах работы гипертекстового интерпретатора и о некоторых улучшениях и дополнениях, которые можно внести в приведенные программы.

## Комментарии к листингам

1. Приведенные в статье программные разработки выполнены на языке программирования C++, хотя, естественно, нет совершенно никаких ограничений на использование других языков: авторы работы [1] создали свою собственную гипертекстовую систему на языке Prolog.

2. Из-за ограничения объема в тексте статьи приведены лишь ключевые моменты программ, основное пояснение алгоритмов перенесено в сами листинги: комментарии составляют до одной третьей всего объема листингов.

3. Константы, общие для всех классов, вынесены в специальный файл HTType.hpp (листинг 2).

4. Практически все описанные классы используют класс Error (листинг 3), обеспечивающий обработку ошибок. В листинге также приведены коды ошибок и тексты, поясняющие сущность той или иной ошибочной ситуации. ■

Литература

1. Д. Гришечкин, В. Оленев, А. Тихонов. Гипертекстовые системы// Монитор.– 1992.– № 5.

Редакция журналов «Монитор» и «Монитор-Аспект» приглашает к сотрудничеству специалистов, готовых предложить качественные обзоры ситуации в различных секторах российского рынка программ и компьютеров. Особый интерес представляют следующие темы:
- рабочие станции;
- САПР;
- компиляторы;
- мощные многоплатформные СУБД;
- анимация.

Приглашаются также опытные программисты и электронщики для участия в тестировании программных продуктов и аппаратуры.

Тел. 335-68-31, 238-99-33. Дмитрий Бройтман

Монитор — 1993

Подписной купон

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|

МОНИТОР

*Адрес* _____

*Ф.И.О. или название организации* _____

Стоимость подписки на 1 номер журнала "Монитор" — 250 рублей. В 1993 году будет выпущено 8 номеров (годовая подписка — 2000 руб.). Отметьте крестиками те номера журнала, которые Вы хотите получить. Перечислите сумму оплаты на счет НТО "Софт-Москва": р/с 467002 в КБ "Банк развития XXI век", к/с 161816 в РКЦ ГУ ЦБ РФ по г. Москве, МФО 201791.
Вышлите подписной купон и квитанцию об оплате по адресу: 117419 Москва, а/я 765.

### Листинг 1а. Файл HTInput.hpp

```
// Описание класса HTInput - поддержка потока входных данных
#ifndef _HTINPUT_
#define _HTINPUT_
#include <Error.hpp>
#include <HTType.hpp>
#include <stdio.h>

class HTInput {
      char *FName;            // Имя входного файла данных
      Error      *ErrStream;  // Указатель потока обработки ошибок
      FILE       *InStream;   // Входной файл
      char Buf[255];          // Служебный буфер
public:
      int MaxStrLen;          // Максимально допустимая длина
                              // строки исходного текста
      int MaxKeyWord;         // Максимально допустимая длина
                              // ключа - заголовка темы/ссылки
          // Конструктор
          HTInput(char *fname, int maxstrlen, int maxkeyword, Error *E);
          ~HTInput();  // Деструктор
      char* ReadString(char *s, int StrNum); // Чтение строки исходного текста
      // Выделение слова (заголовка темы/ссылки) из строки
      char* GetWord(const char *s, int StrNum, char *ret);
      // Удаление из строки начальных и концевых пробелов
      char* ShortenWord(const char *s, char *ret);
      int CheckNetName(char *s);  // Проверка на заголовок
                                  // гипертекстовой сети
      int CheckComment(char *s);  // Проверка на комментарий
      int CheckTopic(char *s);    // Проверка на заголовок темы
      int CheckReference(char *s, int Position);    // Проверка на ссылку
};
#endif
```

### Листинг 1б. Файл HTINput.cpp

```
// Класс HTInput - поддержка потока входных данных
#include <HTInput.hpp>
#include <HTType.hpp>
#include <Error.hpp>
#include <string.h>
#include <stdio.h>

//::::::::::::::::::::::::::::::::::::::::::: Конструктор ::::::::::::::::::::::::::::::::::::::::::
HTInput::HTInput(char *fname, int maxstrlen, int maxkeyword, Error *err)
// fname - имя входного файла
// maxstrlen - максимальная длина строки исходного текста
// maxkeyword - максимальная длина заголовка темы/ссылки
// err - поток обработки ошибок
{ FName = new char[strlen(fname)+1]; // Сохраним имя входного файла
      strcpy(FName,strupr(fname));
      MaxStrLen = (maxstrlen > 0)? maxstrlen:0;   // Установим другие
                                                  // параметры
      MaxKeyWord = (maxkeyword > 0)? maxkeyword:0;
      ErrStream = err;
      if((InStream = fopen(FName,"rt")) == NULL) // Откроем входной файл
          ErrStream-> HandleError(EmOPENIN,0,0,STOP);
      if((setvbuf(InStream,NULL,_IOFBF,1024*5)) != 0)     // Установим
                                                          // буферизацию
          ErrStream-> HandleError(EmNOEMEM,0,0,STOP);
}

//::::::::::::::::::::::::::::::::::::::::::: Деструктор ::::::::::::::::::::::::::::::::::::::::::
HTInput::~HTInput()
{ fclose(InStream); delete FName; }

//::::::::::::::::::::::::::::::::::::::::::: Функции - члены ::::::::::::::::::::::::::::::::::::::::::
//.......................................... ReadString ..........................................
char* HTInput::ReadString(char *s, int StrNum)
// Функция читает строку из входного файла
// Возвращает эту строку или NULL в случае конца файла
{ char *Check = fgets(Buf,sizeof(Buf),InStream); // Прочтем строку
      if(strlen(Buf) > MaxStrLen)          // Проверим длину
          ErrStream-> HandleError(EmLINETOOLONG,StrNum,0,GOON);
      if(Check == NULL) strset(s,0);     // Если обнаружился конец файла
      else strcpy(s,Buf);
      return Check;
}

//.......................................... GetWord ..........................................
char* HTInput::GetWord(const char *s, int StrNum, char *r)
// Функция выделяет слово r (заголовок темы/ссылки) из строки s
// Возвращает найденное слово
{ int i = 0, ErrFlag = 0;
      while( !(s[i] == '\0' || s[i] == '\n' ||   // Определим, где начинается
                                                 // слово
               (s[i] == MARKREF && s[i+1] != MARKREF))) i++;
      if(i >= MaxKeyWord)     // Если его длина превышает максимум,
      { i = MaxKeyWord-1;
          ErrFlag = 1; }      // установим флаг ошибки,
      strncpy(Buf,s,i);       // скопируем слово в буфер
      Buf[i] = '\0';
      if(Buf) strcpy(r,Buf);  // Если буфер не пуст
      if(ErrFlag)             // Ошибка!
      ErrStream-> HandleError(EmLONGNAM,StrNum,Buf,GOON);
      return(Buf);
}

//.......................................... ShortenWord ..........................................
char *HTInput::ShortenWord(const char *s, char *ret)
// Функция удаляет из строки s начальные и концевые пробелы
// и помещает ее в ret. Возвращает ret
{ int begin = 0;              // Начало строки
      int end = strlen(s)-1;  // Конец строки
      while(s[begin] == ' ') begin++;          // Удаляем начальные пробелы
      while(s[end] == ' ' && begin < end) end--;    // и концевые
      for(int i = 0; i < end-begin+1; i++)
          ret[i] = s[begin+i];                     // Скопируем строку
      ret[i] = '\0';    return ret;
}

//.......................................... CheckReference ..........................................
int HTInput::CheckReference(char *s, int Pos)
// Функция проверяет, не определена ли ссылка в строке s,
// начиная с позиции Pos. Если да, возвращает смещение от начала
// строки, где определена ссылка, если нет, 0
{ while(s[Pos] != '\0' && s[Pos] != '\n') // Пройдем по всей строке
      { if(s[Pos] == MARKREF)             // Если найден символ "&",
              if(s[Pos+1] != MARKREF) return Pos+1;    // вернем значение
              else if(s[Pos+1] == MARKREF) Pos++;      // Обработка
                                                       // ситуации "&&"
          Pos++;
      }
      return 0;       // Ссылка не найдена
}

//.......................................... CheckNetName ..........................................
int HTInput::CheckNetName(char *s)
// Функция делает проверку на заголовок гипертекстовой сети
{ return !(strnicmp(s,MARKTITLE,strlen(MARKTITLE))); }

//.......................................... CheckComment ..........................................
int HTInput::CheckComment(char *s)
// Функция делает проверку на комментарий
{ return s[0] == MARKCOMMENT; }

//.......................................... CheckTopic ..........................................
int HTInput::CheckTopic(char *s)
// Функция делает проверку на определение заголовка темы
{ return s[0] == MARKTOPIC; }
```

### Листинг 2. Файл HTType.hpp

```
// HTType.hpp - описание различных констант,
// которые используются как в трансляторе, так и в интерпретаторе
#ifndef _HTTYPE_
#define _HTTYPE_

//.......................... Метки, используемые в исходном тексте ..........................
#define MARKTITLE "__TITLE:"  // Метка заголовка гипертекста
#define MARKSPECIAL     '^'   // Метка определения
                              // спецсимвола/атрибута: ^Cxx, ^Axx
#define MARKTOPIC       '|'   // заголовка темы:      |topic
#define MARKREF         '&'   // ссылки:     &topic
#define MARKCOMMENT     ';'   // комментария:  ;comment
#define MARKATTRIB      'A'   // атрибута:   ^Axx
#define MARKCHAR        'C'   // спецсимвола:   ^Cxx

#define MARKC1          '1'   // Метки предопределенных атрибутов
#define MARKC2          '2'   // ^1 ... ^5
#define MARKC3          '3'
#define MARKC4          '4'
#define MARKC5          '5'

//.......................... Метки, используемые в базе данных ..........................
// Внимание: не использовать метки 0A,0D - это управляющие символы
// Метка файла базы данных
#define DBFILE_ID \
    "\x1AHyperText :: CopyRight (C) '93 by Monitor & Stan Kurkovsky"
#define COMPRESS        '\x2'   // Метка - сжатые данные
#define TYPEREF         '\x3'   // ссылка
#define TYPESPECIAL     '\x4'   // спецсимвол/атрибут
#define TYPEATTRIB      '\x5'   // атрибут
#define TYPECHAR        '\x6'   // спецсимвол
#define SYNCH_BYTE      '\x7'   // начало текста темы
#define TYPEC1          '\x10'  // Метка предопределенных атрибутов
#define TYPEC2          '\x11'  // ^1 ... ^5
#define TYPEC3          '\x12'
#define TYPEC4          '\x13'
#define TYPEC5          '\x14'
#define DELAY 128               // Значение, прибавляемое к счетчику повторений,
                                // чтобы отличить его от прочих меток
#endif
```

### Листинг 3а. Файл Error.hpp

```
#ifndef _ERROR_
#define _ERROR_

enum MakerErrorCode {                  // Коды ошибок транслятора
      EmOPENIN, EmOPENOUT, EmOPENTMP,EmREADIN,
      EmWRTMP, EmDISKFULL, EmNOEMEM,EmLINETOOLONG,
      EmLONGNAM, EmREFITSLF, EmREFNOFD, EmDUPNAM,
      EmLONGENTRY,EmNONAMES, EmOUTTEXT, EmBADCHAR,
      EmTOOMANY};

enum ViewerErrorCode {                 // Коды ошибок интерпретатора
EvSUCCESS = 0, EvOPENDB, EvFILNODB, EvTOOMANYDB,
EvDBREG, EvDBNOTREG, EvTOPNOFD,
EvTBEGNOFD, EvREADCOMP, EvNOEMEM};

enum ProgramState {STOP = 1, GOON = 0}; // Состояние программы
class Error {
public:
      int Total;      // Счетчик ошибок
      int MaxError; // Максимально допустимое количество ошибок
      char *FName; // Имя обрабатываемого файла
          Error(int MaxError, char *FName);     // Конструктор
          ~Error();                             // Деструктор
      // Обработчик ошибок транслятора
      void HandleError(MakerErrorCode, int StrNum, char *Item, ProgramState);
};
#endif
```

### Листинг 3б. Файл Error.cpp

```
// Класс Error - поддержка обработки ошибок
#include <Error.Hpp>
#include <stdio.h>
#include <stdlib.h>      //exit()
#include <string.h>

// Сообщения об ошибках гипертекстового транслятора
char *MakerErrorMessage[] = {
      "Ошибка открытия входного файла",
      "Ошибка открытия выходного файла",
      "Ошибка открытия временного файла",
      "Ошибка чтения",
      "Ошибка записи",
      "Нет свободного места на диске",
      "Недостаточно памяти",
      "Строка слишком длинная",
      "Заголовок темы/ссылки слишком длинный",
      "Ссылка сделана сама на себя",
      "Ссылка не найдена",
      "Повторяющееся имя темы",
      "Тема слишком длинная",
      "Не определено ни одной темы",
      "Текст не принадлежит ни одной теме",
      "Неправильное определение спецсимвола (^C) / атрибута (^A)",
      "Слишком много ошибок"};

// Сообщения об ошибках гипертекстового интерпретатора
char *ViewerErrorMessage[] = {
      "Ошибка открытия базы данных",
      "Файл не является базой данных",
      "Слишком много баз данных",
      "База данных уже зарегистрирована",
      "База данных не зарегистрирована",
      "Тема не найдена в заголовке",
      "Не найдено начало темы",
      "Ошибка чтения/декомпрессии",
      "Недостаточно памяти"};

//:::::::::::::::::::::::::::::::::::::::: Конструктор ::::::::::::::::::::::::::::::::::::::::
Error::Error(int maxerr, char *fname)
{ MaxError = (maxerr > 0)?maxerr:0;
      if(fname! = NULL) {
            FName = new char[strlen(fname) + 1];
            strcpy(FName,strupr(fname));
      }
      else FName = NULL;
      Total = 0;
}

//:::::::::::::::::::::::::::::::::::::::: Деструктор ::::::::::::::::::::::::::::::::::::::::
Error::~Error()
{ if(FName) delete FName; }

//.......................................... HandleError ..........................................
void Error::HandleError(MakerErrorCode ErrCode,int StrNum,
            char *Item, ProgramState State)

// Обработчик ошибок транслятора
// ErrCode - код ошибки в диапазоне enum MakerErrorCode
// StrNum - номер строки, где произошла ошибка (если есть)
// Item - часть текстовой строки, где произошла ошибка (если есть)
// State - дальнейшее состояние программы
// (продолжать/остановиться)
{ Total++;     // Увеличим счетчик ошибок
      printf("#%d Error %s ",Total,FName);
      if(StrNum) printf("%d ",StrNum);
      printf(": %s",MakerErrorMessage[ErrCode]);
      if(Item! = NULL) printf(" \"%s\"",Item);
      printf("\n");
      if(State = = STOP) exit(1);     // Если надо, завершим программу
      // Если превышено максимальное число ошибок
      if(Total> = MaxError) HandleError(EmTOOMANY,0,NULL,STOP);
}
```

### Листинг 4а. Файл HTNode.hpp

```
// Описание класса HTNode - узел гипертекстовой сети
#ifndef _HTNODE_
#define _HTNODE_

class HTNode {
public:
      char *KeyWord;          // Заголовок темы/ссылки
      long Seek;              // Для темы: адрес в файле
      int  Size;              // размер в строках
      int  Appears;           // Номер строки, где в исходном тексте
                              // появляется заголовок темы/ссылка
      HTNode  *NextTopic;     // Указатель на следующую тему в списке
      HTNode  *PrevTopic;     // Указатель на предыдущую тему в списке
      HTNode  *Reference;     // Указатель на следующую ссылку в списке
      int  X,Y;               // Позиция в тексте
                              // (используется только в интерпретаторе)
            // Конструктор
            HTNode(char *keyword, long seek, int size, int appears, int = 0, int = 0);
            ~HTNode();        // Деструктор

      // Переопределенные операторы
      friend int operator > (HTNode &a, HTNode &b); // Сравнение двух
                                                    // объектов
      friend int operator < (HTNode &a, HTNode &b); // по полю KeyWord
      friend int operator > (char *a, HTNode &b);   // Сравнение поля
                                                    // KeyWord
      friend int operator < (char *a, HTNode &b);   // со строкой
};
```

### Листинг 4б. Файл HTNode.cpp

```
// Класс HTNode - узел гипертекстовой сети
#include <HTNode.hpp>
#include <string.h>

//:::::::::::::::::::::::::::::::::::::::: Конструктор ::::::::::::::::::::::::::::::::::::::::
HTNode::HTNode(char *keyword, long seek, int size, int appears, int x, int y)
{ NextTopic = PrevTopic = NULL;  // Указатели будут инициализированы
      Reference = NULL;          // объектами классов верхнего уровня
      if(keyword)
      { KeyWord = new char[strlen(keyword) + 1]; // Создание
            strcpy(KeyWord,strupr(keyword));     // и заполнение поля ключа
      }
      Seek = seek; Size = size;                  // Заполнение
      Appears = appears; X = x; Y = y;           // других полей
}

//:::::::::::::::::::::::::::::::::::::::: Деструктор ::::::::::::::::::::::::::::::::::::::::
HTNode::~HTNode()
{ if(KeyWord) delete KeyWord; }

//.......................................... CompareStr ..........................................
int CompareStr(const char *s1, const char *s2)
// Функция сравнения двух строк, аналог strcmp,
// здесь приводится исключительно для наглядности
{ for(; *s1 = = *s2; s1++, s2++) if(*s1 = = '\0') return 0;
      return *s1 - *s2;
}

//:::::::::::::::::::::::::::::::::::::::: Операторы ::::::::::::::::::::::::::::::::::::::::
int operator > (HTNode &a, HTNode &b)
{ return ( (CompareStr(a.KeyWord,b.KeyWord)) > 0); }

int operator < (HTNode &a, HTNode &b)
{ return ( (CompareStr(a.KeyWord,b.KeyWord)) < 0); }

int operator > (char *a, HTNode &b)
{ return ( (CompareStr(a,b.KeyWord)) > 0); }

int operator < (char *a, HTNode &b)
{ return ( (CompareStr(a,b.KeyWord)) < 0); }
```

### Листинг 5а. Файл HTNet.hpp

```
// Описание класса HTNet - создание и поддержка гипертекстовой сети
#ifndef _HTNET_
#define _HTNET_

#include <HTNode.Hpp>
#include <Error.Hpp>
// Константы, возвращаемые логическими функциями
// проверок и поиска
enum CheckState { Success = 1, Failure = 0 };

class HTNet {
      HTNode  *LastTopic;     // Указатель на текущую (последнюю) тему
      HTNode  *LastReference; // Указатель на текущую (последнюю) ссылку
      Error   *ErrStream;     // Указатель потока обработки ошибок
      int   MaxTopicSize;// Максимально допустимый размер темы (в строках)
public:
      unsigned  TotalTopics;        // Количество тем в сети
      unsigned  TotalReferences;    // Количество ссылок в сети
      char      *NetTitle;          // Заголовок гипертекстовой сети
      HTNode  *Begin;               // Указатель на начало списка (сети)
            HTNet(Error *, int);    // Конструктор
            ~HTNet();               // Деструктор
      // Создание новой темы
      void NewTopic(char *Name, long seek, int appears);
      void NewReference(char *Name, int appears); // Создание новой ссылки
      void IncreaseTopicSize(int StrNum); // Увеличение поля размера темы
      void RegisterNetTitle(char *Name); // Сохранение заголовка сети
      CheckState AnalyseNet();      // Функция анализа корректности сети
      // Функция анализа корректности ссылок
      CheckState AnalyseReferences(HTNode *Reference, HTNode *Parent);
      // Функция нахождения данной темы в сети (для проверки)
      CheckState InTopicList(char *Name);
};
```

### Листинг 5б. Файл HTNet.cpp

```
// Класс HTNet - создание и поддержка гипертекстовой сети
#include <HTNet.Hpp>
#include <Error.Hpp>
#include <HTType.Hpp>
#include <string.h>

//:::::::::::::::::::::::::::::::::::::::: Конструктор ::::::::::::::::::::::::::::::::::::::::
HTNet::HTNet(Error *Err, int maxtopicsize)
// Err - поток обработки ошибок
// maxtopicsize - макс. размер текста темы в строках
{ Begin = LastTopic = LastReference = NULL;
      NetTitle = NULL;    ErrStream = Err;
      MaxTopicSize = maxtopicsize;
      TotalTopics = TotalReferences = 0;
}

//:::::::::::::::::::::::::::::::::::::::: Деструктор ::::::::::::::::::::::::::::::::::::::::
HTNet::~HTNet()
{ HTNode *TmpTopic = Begin;
      while(TmpTopic! = NULL)                 // По списку всех тем ...
      { HTNode *TmpRef = TmpTopic-> Reference;
            while(TmpRef! = NULL)             // По списку всех ссылок
            { HTNode *TmpDel = TmpRef;        // запомним следующую
                  TmpRef = TmpRef-> Reference; delete TmpDel;
                                              // Уничтожим ссылку
            }
            HTNode *TmpDel = TmpTopic;        // Запомним следующую
            TmpTopic = TmpTopic-> NextTopic;
            delete TmpDel;                    // Уничтожим тему
```

```
    }
    if(NetTitle) delete NetTitle;
}
//::::::::::::::::::::::::::::::::::::::::: Функции - члены ::::::::::::::::::::::::::::::::::::::::
//..... RegisterNetTitle .....
void HTNet::RegisterNetTitle(char *Name)
// Функция регистрирует заголовок Name гипертекстовой сети
{ NetTitle = new char[strlen(Name)+1];
    strcpy(NetTitle,Name+strlen(MARKTITLE));
    if(NetTitle[strlen(NetTitle)-1] = = '\n') NetTitle[strlen(NetTitle)-1] = '\0';
}
//..... NewTopic .....
void HTNet::NewTopic(char *Name, long seek, int StrNum)
// Функция создает новый узел (тему) гипертекстовой сети
{ // Проверим, не зарегистрирована ли уже такая тема
    if(InTopicList(strupr(Name))! = Success)
    { ErrStream-> HandleError(EmDUPNAM,StrNum,Name,GOON); return; }
    HTNode *NewNode = new HTNode(Name,seek,0,StrNum);
    if(Begin = = NULL) Begin = NewNode;          // Изменим головной узел
    else LastTopic-> NextTopic = NewNode;        // или последний
    LastReference = NULL; LastTopic = NewNode; TotalTopics++;
}
//..... NewReference .....
void HTNet::NewReference(char *Name,int StrNum)
// Функция создает новый узел (ссылку) гипертекстовой сети
{ HTNode *NewNode = new HTNode(Name,0,0,StrNum);
    if(LastReference = = NULL) LastTopic-> Reference = NewNode;
    else LastReference-> Reference = NewNode;
    LastReference = NewNode; TotalReferences++;
}
//..... IncreaseTopicSize .....
void HTNet::IncreaseTopicSize(int StrNum)
// Функция увеличивает поле размера текущей темы и делает проверки
{ if(LastTopic = = NULL)      // Если список тем пуст,
    { // следовательно, текст не принадлежит ни одной теме
        ErrStream-> HandleError(EmOUTTEXT,StrNum,NULL,GOON);
        return; }
    LastTopic-> Size++;
    if(LastTopic-> Size = = MaxTopicSize)  // Если превышен
                                           // максимальный размер
        ErrStream-> HandleError(EmLONGENTRY,StrNum,NULL,GOON);
}
//..... AnalyseNet .....
CheckState HTNet::AnalyseNet()
// Функция анализирует гипертекстовую сеть
{ HTNode *Current = Begin;              // Текущий проверяемый узел
    CheckState RetValue = Success;      // Текущее состояние проверки
    if(Current = = NULL)                // Если сеть пуста
    { ErrStream-> HandleError(EmNONAMES,0,NULL,GOON);
        RetValue = Failure;
    }
    while(Current! = NULL)              // Пройдем по всей сети
    { // Если список ссылок не пуст, проверим его
        if(Current-> Reference! = NULL)
        if(AnalyseReferences(Current-> Reference,Current) = = Failure)
            RetValue = Failure;
        Current = Current-> NextTopic;
    }
    return RetValue;
}
//..... AnalyseReferences .....
CheckState HTNet::AnalyseReferences(HTNode *Ref, HTNode *Parent)
// Функция рекурсивно анализирует ветвь ссылок от темы Parent,
// начиная со ссылки Ref
{ HTNode *Current = Begin;
    CheckState RetValue = Success, AddValue = Success;
    // Пройдем по списку тем гипертекстовой сети и попытаемся найти
    // тему, на которую указывает ссылка
    while((strcmp(Current-> KeyWord,Ref-> KeyWord)) &&
        (Current! = NULL))
        Current = Current-> NextTopic;
    if(!Current)    // Такая тема не найдена -> ошибка
    { ErrStream-> HandleError(EmREFNOFD,Ref-> Appears,
                                Ref-> KeyWord,GOON);
        RetValue = Failure;
    } else          // Такая тема найдена
        if(!strcmp(Parent-> KeyWord,Ref-> KeyWord))
        { // Если это ссылка из темы Parent, то есть ссылка сама на себя
            ErrStream-> HandleError(EmREFITSLF,Ref-> Appears,
                                    Ref-> KeyWord,GOON);
            RetValue = Failure;
        }
    // Проанализируем следующую ссылку в списке
    if(Ref-> Reference! = NULL)
        AddValue = AnalyseReferences(Ref-> Reference,Parent);
    if(RetValue = = Success && AddValue = = Failure) RetValue = Failure;
    return RetValue;
}
//..... InTopicList .....
CheckState HTNet::InTopicList(char *Name)
// Функция нахождения данной темы в сети
{ HTNode *Current = Begin;
    while(strcmp(Name,Current-> KeyWord) && Current! = NULL)
        Current = Current-> NextTopic;
    if(Current! = NULL) return Failure;
    else return Success;
}
```

■

### Листинг 6а. Файл HTOutput.hpp

```
// Определение класса HTOutput - поддержка выходного и
// временного потоков данных
#ifndef _HTOUTPUT_
#define _HTOUTPUT_
#include <HTNet.hpp>
#include <Error.hpp>
#include <stdio.h>
class HTOutput {
    char *OutFName;         // Имя выходного файла
    char *TmpFName;         // Имя временного файла
    FILE    *OutStream;     // Выходной файл
    FILE    *TmpStream;     // Временный файл
    Error   *ErrStream;     // Поток обработки ошибок
public:
        HTOutput(char *fname, Error *E);    // Конструктор
        ~HTOutput();                        // Деструктор
    int Hex(char a, char b); // Составляет шестнадцатиричное число из
                             // двух символов
    // Запись строки во временный поток
    void WriteString(char *s, int StrNum, int NewTopicFlag);
    void CompressLine(char *s, int StrNum);     // Сжатие строки
    void CreateDataBase(HTNet *Net);            // Создание базы данных
    // Определение текущей позиции во временном файле
    unsigned long GetSeek();
};
#endif
```

■

### Листинг 6б. Файл HTOutput.cpp

```
// Класс HTOutput - поддержка временного и выходного потоков
#include <HTOutput.hpp>
#include <HTNet.hpp>
#include <HTType.hpp>
#include <Error.hpp>
#include <string.h>
#include <ctype.h>

// :::::::::::::::::::::::::::::::::::::::: Конструктор ::::::::::::::::::::::::::::::::::::::::
HTOutput::HTOutput(char *fname, Error *Err)
// fname - имя выходного файла (базы данных)
// Err - поток обработки ошибок
{ ErrStream = Err;
    OutFName = new char[strlen(fname)+1];    // Сохраним имя файла
    strcpy(OutFName,strupr(fname));
    if((OutStream = fopen(OutFName,"wb")) = = NULL)    // Открытие файла
        ErrStream-> HandleError(EmOPENOUT,0,OutFName,STOP);
    if((setvbuf(OutStream,NULL,_IOFBF,1024*5))! = 0)  // Установка
                                                      // буферизации
        ErrStream-> HandleError(EmNOEMEM,0,0,STOP);
    TmpFName = new char[12];            // Создание и сохранение имени
    tmpnam(TmpFName);                   // временного файла
    if((TmpStream = fopen(TmpFName,"wb")) = = NULL)   // Открытие файла
        ErrStream-> HandleError(EmOPENTMP,0,OutFName,STOP);
    if((setvbuf(TmpStream,NULL,_IOFBF,1024*5))! = 0)  // Установка
                                                      // буферизации
        ErrStream-> HandleError(EmNOEMEM,0,0,STOP);
}
// :::::::::::::::::::::::::::::::::::::::: Деструктор ::::::::::::::::::::::::::::::::::::::::
HTOutput::~HTOutput()
{ fclose(OutStream);            // Закрытие выходного файла
    fclose(TmpStream);          // Закрытие временного файла
    unlink(TmpFName);           // Его уничтожение
    delete OutFName;  delete TmpFName;
}
// :::::::::::::::::::::::::::::::::::::::: Функции - члены ::::::::::::::::::::::::::::::::::::::::
//..... Hex .....
int HTOutput::Hex(char a, char b)
// Функция составляет из символов a и b 16-ричное число,
// с проверкой осуществимости
// Возвращает      > = 0 в случае успеха,
//                 -1 в случае ошибки
{ int LowByte, HighByte;
    a = toupper(a); b = toupper(b);
    if(a > = '0' && a < = '9') HighByte = a-'0';
    else if(a > = 'A' && a < = 'F') HighByte = a-'A'+10;
        else return -1;
    if(b > = '0' && b < = '9') LowByte = b-'0';
    else if(b > = 'A' && b < = 'F') LowByte = b-'A'+10;
        else return -1;
    return (HighByte*16+LowByte);
}
//.....WriteString.....
void HTOutput::WriteString(char *s, int StrNum, int NewTopicFlag)
// Функция записывает строку s в выходной файл,
// в случае, если эта строка - первая в теме, о чем сигнализирует
// NewTopicFlag, ставит метку начала темы SYNCH_BYTE
{ CompressLine(s,StrNum);           // Сначала - упаковка строки
    if(NewTopicFlag) fputc(SYNCH_BYTE,TmpStream);
    if(fputs(s,TmpStream) = = EOF)  // Запишем строку в файл
        ErrStream-> HandleError(EmWRTMP,StrNum,0,STOP);
}
//.....CompressLine.....
void HTOutput::CompressLine(char *s, int StrNum)
// Функция упаковывает строку s, выставляя при этом метки
// ссылок, спецсимволов и атрибутов.
// Упакованная строка помещается обратно в s.
{ char buf[255];    // Буфер для хранения сжатой строки
    int i,j;        // i - индекс исходной строки, j - сжатой
```

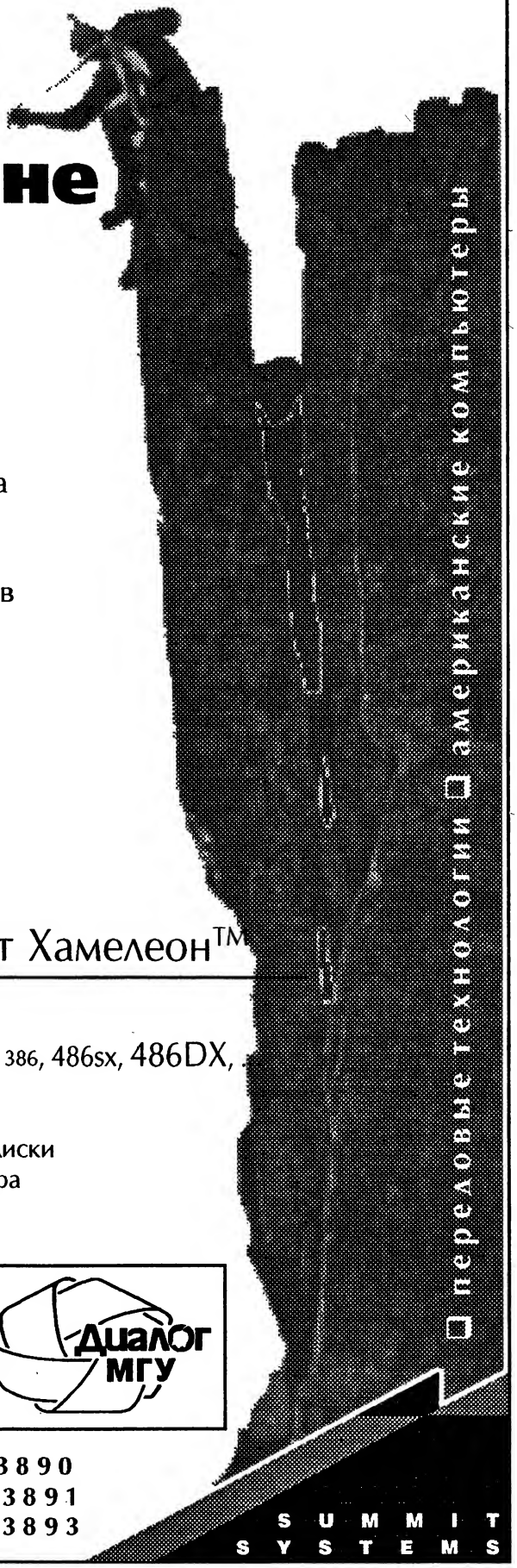
### Листинг 6б (продолжение)

```
    int LengthOfStr = strlen(s);
    // Здесь удаляются все пробелы в конце строки (за ненадобностью)
    while(s[LengthOfStr-2] = = ' ' && LengthOfStr > = 2) LengthOfStr--;
    s[LengthOfStr-1] = '\n'; s[LengthOfStr] = '\0';
    for(i = 0, j = 0; i < LengthOfStr; i++)      // Цикл по символам строки
    { if(s[i] = = MARKREF)                       // Если стоит метка ссылки "&"
        { if(s[i+1] = = MARKREF)                 // Ситуация "&&":
            { buf[j] = MARKREF; i++;             // оставим только 1 символ
            } else buf[j] = TYPEREF; // Ситуация "&":
            j++;                                 // выставим метку
        } else if(s[i] = = MARKSPECIAL)  // Если стоит метка "^"
        { if(s[i+1] = = MARKSPECIAL)     // Ситуация "^^":
            { buf[j] = MARKSPECIAL;      // оставим только 1 символ
              i++; j++; continue;
            }
            buf[j] = TYPESPECIAL;  // Ситуация "^": выставим метку
            switch(toupper(s[i+1])) // Какой следующий символ?
            {case MARKC1: buf[j+1] = TYPEC1; break; // Ситуация "^1"
            case MARKC2: buf[j+1] = TYPEC2; break;  // Ситуация "^2"
            case MARKC3: buf[j+1] = TYPEC3; break;  // Ситуация "^3"
            case MARKC4: buf[j+1] = TYPEC4; break;  // Ситуация "^4"
            case MARKC5: buf[j+1] = TYPEC5; break;  // Ситуация "^5"
                                                    // Ситуация "^Axx"
            case MARKATTRIB: buf[j] = TYPEATTRIB; goto ProcessChar;
                                                    // Ситуация "^Cxx"
            case MARKCHAR: buf[j] = TYPECHAR;
            // Конечно, нехорошо использовать метки, но иногда
            // очень хочется ...
            ProcessChar:  // Проверка, подходят ли символы
            if(!isxdigit(s[i+2]) || !isxdigit(s[i+3]))
            ErrStream->HandleError(EmBADCHAR,StrNum,NULL,GOON);
            // Преобразование в число
            else buf[j+1] = Hex(s[i+2],s[i+3]);
            i+ = 2; break;
            // А здесь нам попалось вообще неизвестно что - это ошибка
            default: ErrStream->HandleError(EmBADCHAR,
                                                    StrNum,NULL,GOON);
            break;
            }
            j+ = 2; i++;
        }
    // Обнаружились повторяющиеся символы, к тому же их там больше
    // трех
    else if((s[i] = = s[i+1]) && (s[i+1] = = s[i+2]))
    { char Count = 3;                            // Это счетчик повторений
    while(s[i+Count] = = s[i]) Count++;  // Здесь его будут увеличивать
    buf[j] = COMPRESS;                           // Поставим метку сжатия
    buf[j+1] = Count+DELAY;    // Изменим счетчик на константу, чтобы
    // избежать его совпадения со всяческими управляющими символами,
    // как то:      0x0A (LF) - line feed
    //              0x0D (CR) - carriage return
            buf[j+2] = s[i]; j+ = 3; i+ = Count-1;
        } else      // Здесь обработаем все остальные символы -
        {           // просто перенесем их из одной строки в другую
            buf[j] = s[i]; j++;
        }
    }
    // Если случилось так, что результирующая строка пустая,
    // добавим туда символ новой строки
    if(j = = 0) buf[j++] = '\n';
    // А если и такого символа там нет
    else if(buf[j-1] ! = '\n') buf[j++] = '\n';
    buf[j] = '\0';   // Выставим завершающий ноль
    strcpy(s,buf);
}

//.....CreateDataBase.....
void HTOutput::CreateDataBase(HTNet *Net)
// Функция создает базу данных,
// используя информацию
// из гипертекстовой сети Net
{ fclose(TmpStream);          // Временный файл был открыт для записи,
    TmpStream = fopen(TmpFName,"rb");        // откроем его для чтения
    fprintf(OutStream,"%s%c",DBFILE_ID,0);  // Запишем метку
                                             // идентификации
    fprintf(OutStream,"0000");  // Это делается только для того, чтобы
                                // сохранить место, зачем - смотри далее
    putw(Net->TotalTopics,OutStream);   // Запишем количество тем
    // Если определен заголовок сети, запишем его
    if(Net->NetTitle! = NULL) fprintf(OutStream,"%s%c",Net->NetTitle,0);
    else fputc(0,OutStream);
    // Пройдемся по гипертекстовой сети, точнее - по темам
    for(HTNode *Current = Net->Begin; Current! = NULL;
Current = Current->NextTopic)
    {       // Запишем длину заголовка темы, сам заголовок
            // и размер темы в строках
    fprintf(OutStream,"%c%s%c",
        strlen(Current->KeyWord),Current->KeyWord,Current->Size);
    putw(Current->Seek / 0xFF, OutStream);   // Так записывается
                                             // 4-байтовое смещение
    putw(Current->Seek % 0xFF, OutStream);  // от начала файла
    }
    // Сохраним текущее положение файла -
    // именно здесь начинаются тексты тем
        unsigned long DataBeginAddr = ftell(OutStream);
        fseek(OutStream,strlen(DBFILE_ID)+1,SEEK_SET);
        putw(DataBeginAddr / 0xFF, OutStream);// Запишем это значение
                                              // в специально
        putw(DataBeginAddr % 0xFF, OutStream);  // отведенное место,
                                              // смотри ранее,
        fseek(OutStream,DataBeginAddr,SEEK_SET);// и вернемся назад
    // Процедура копирования временного файла в выходной
    // Все очень просто - без комментариев
    unsigned CopyBufSize = 1024*5, BytesRead = 1, BytesWrite;
    char *CopyBuf =  new char[CopyBufSize];
    while(BytesRead)
    { BytesRead = fread(CopyBuf,1,CopyBufSize,TmpStream);
        BytesWrite = fwrite(CopyBuf,1,BytesRead,OutStream);
        if (BytesRead! = BytesWrite)
          { delete CopyBuf;
ErrStream->  HandleError(EmDISKFULL,0,NULL,STOP); }
          }
    delete CopyBuf;
}

//.....GetSeek.....
unsigned long HTOutput::GetSeek()
// Функция сообщает текущую позицию в файле
{ return ftell(TmpStream); }
```

### Листинг 7. Файл HTMake.cpp

```
// Гипертекстовый транслятор
#include <Error.hpp>
#include <HTInput.hpp>
#include <HTOutput.hpp>
#include <HTNet.hpp>
#include <stdio.h>
#include <stdlib.h>      // atexit()

Error       *ErrStream = NULL;
HTInput     *InStream = NULL;
HTOutput    *OutStream = NULL;
HTNet       *Net = NULL;

int MakeHyperText(char *SourceFile, char *DestFile)
{ char StringBuf[255];  // Сюда будем читать строку
    ErrStream = new Error(25,SourceFile);
    InStream = new HTInput(SourceFile,255,30,ErrStream);
    OutStream = new HTOutput(DestFile,ErrStream);
    Net = new HTNet(ErrStream,255);
    printf("Чтение исходного текста %s:\n",SourceFile);
    int NewTopicFlag = 0;
    for( int CurrentString = 1;
        InStream->ReadString(StringBuf,CurrentString); CurrentString++)
    { char buf[50];      // Буфер для заголовков тем/ссылок
        if(InStream->CheckComment(StringBuf)) continue;// Комментарий
        // Заголовок гипертекстовой сети
        if(CurrentString = = 1 && InStream->CheckNetName(StringBuf))
        { InStream->ShortenWord(StringBuf,buf);
            Net->RegisterNetTitle(buf);
            continue;
        }
        if(InStream->CheckTopic(StringBuf)) // Заголовок новой темы
        { InStream->ShortenWord(InStream->GetWord
                                    (StringBuf+1,CurrentString,buf),buf);
            Net->NewTopic(buf,OutStream->GetSeek(),CurrentString);
            NewTopicFlag = 1;
            continue;
        }
        int RefPosition = 0;  // Поиск ссылок
        while((RefPosition = InStream->CheckReference
                                    (StringBuf,RefPosition))! = 0)
        { InStream->GetWord(StringBuf+RefPosition,CurrentString,buf);
            RefPosition+ = strlen(buf)+1;
            InStream->ShortenWord(buf,buf);
            Net->NewReference(buf,CurrentString);
        }
        Net->IncreaseTopicSize(CurrentString);
        OutStream->WriteString(StringBuf,CurrentString,NewTopicFlag);
        NewTopicFlag = 0;
    }
    if(ErrStream->Total! = 0) exit(1); // Были ошибки
    printf("Анализ гипертекстовой сети %s:\n",SourceFile);
    if((Net->AnalyseNet()) = = Success)
    { printf("Создание базы данных %s:\n",DestFile);
        OutStream->CreateDataBase(Net);
        printf("\n%d строк. Темы: %d, Ссылки: %d\n",
            CurrentString-1,Net->TotalTopics,Net->TotalReferences);
        return 1;
    } else      // Были ошибки
    { printf("%d ошибок\n",ErrStream->Total); return 0; }
}


// Функция уничтожает все объекты
void Terminate(void)
{ if(Net) delete Net; if(InStream) delete InStream;
    if(OutStream) delete OutStream;
    if(ErrStream) delete ErrStream;
}


int main(int argc, char *argv[])
{ printf("Гипертекстовый транслятор.\n");
    if(argc! = 3)
    { printf("Использование: HTMAKE SourceFile DataBase\n");
        return 1;
    }
    atexit(Terminate);
    MakeHyperText(argv[1],argv[2]);
    return 0;
}
```

# Вместе с Вами к Вершине




Самые передовые американские технологии **CHIPS & Technologies** позволят Вам всегда оставаться на вершине быстро меняющегося рынка микропроцессоров. Современный **модульный** дизайн наших компьютеров обеспечивает уникальную возможность приобрести Ваш следующий, самый мощный и самый современный компьютер по цене недорогого процессорного модуля.

## Компьютер Саммит Хамелеон™

- ❑ Рабочая Windows - станция
- ❑ Сменные процессорные модули 386, 486sx, 486DX,
- ❑ Акселератор Wingine
- ❑ SuperVGA монитор
- ❑ Высокоскоростные жесткие диски
- ❑ Русифицированная клавиатура
- ❑ Лицензионная ДОС
- ❑ **2 года гарантии**

**Законченные решения на базе семейства компьютеров Саммит Системс.**

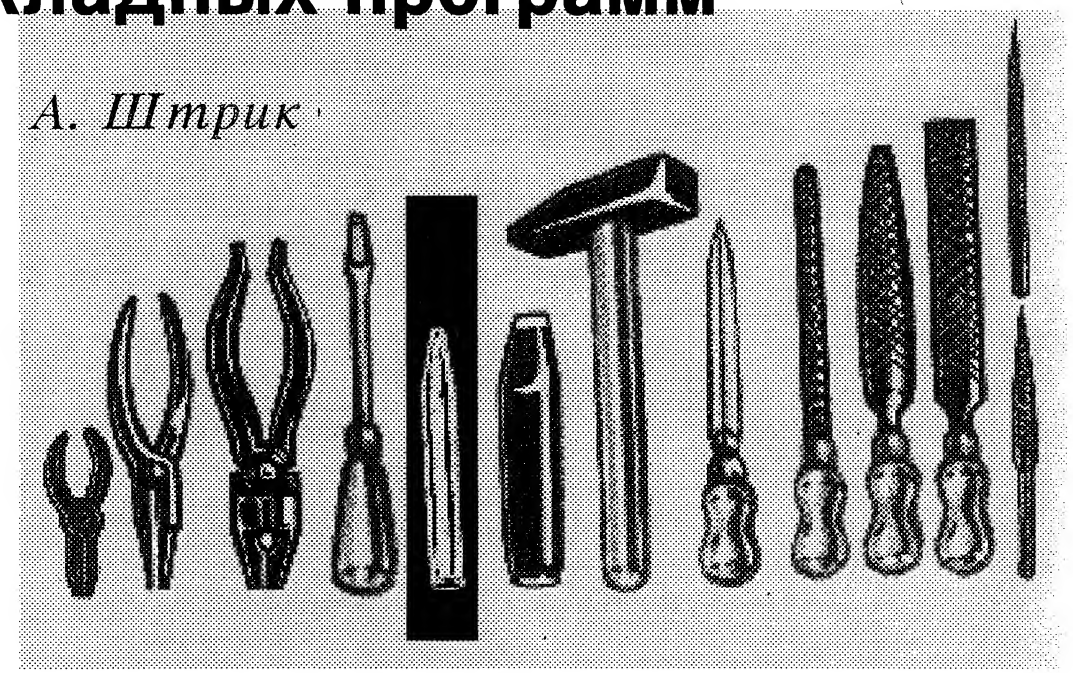
Диалог МГУ

телефоны (095) 939-3890
(095) 939-3891
факс (095) 939-3893

SUMMIT SYSTEMS

# Отечественные инструментальные средства автоматизации создания прикладных программ

А. Штрик



В статьях [1], [2] была рассмотрена современная концепция построения инструментальных CASE (Computer Aided Software Engineering) средств автоматизированного создания прикладного программного обеспечения (ПО), а также приведено краткое описание характеристик и функциональных возможностей зарубежных CASE средств, официально распространяемых в нашей стране. В настоящей статье описывается текущее состояние отечественных инструментальных средств автоматизации создания прикладного ПО. Сначала приводятся интегральные характеристики, полученные по всей известной совокупности отечественных средств, что позволяет сравнивать уровни развития наших и зарубежных CASE средств. Затем даются перечень и характеристики коммерчески доступных отечественных инструментальных средств, созданных за два последних года (в основном по материалам выставок Софтул-92 и Информатика-92); эти данные позволят заинтересованным читателям оценить возможности применения перечисленных отечественных CASE средств в своих разработках. Наконец, в заключении статьи рассмотрены некоторые проблемы отечественных CASE средств, а также возможные тенденции в их развитии и применении.

## Интегральные характеристики отечественных инструментальных средств

Отечественные инструментальные средства разработки ПО, в соответствии со сложившейся у нас классификацией, представляют следующие уровни интеграции:

- автономные инструментальные CASE средства, то есть те, которые поддерживают только отдельные операции и работы по созданию ПО (редакторы текстов и программ, средства создания документации, отладчики и так далее);
- системы программирования на языках третьего и четвертого поколения — средства, включающие компилятор и некоторый минимум программ, которые обеспечивают поддержку части жизненного цикла, начиная от этапа написания исходного текста программы до получения тестированной и документированной машинной программы;
- интегрированные инструментальные CASE средства, предназначенные для эффективной поддержки процессов создания прикладного ПО на всех или на большинстве этапов жизненного цикла;

– интегрированные пакеты, которые, имея собственное функциональное назначение в той или иной прикладной области и работая на определенных системно-аппаратных платформах, в то же время применяются в качестве проблемно-ориентированных инструментальных средств для создания ПО в этой конкретной прикладной области.

В процессе анализа состояния отечественного программного обеспечения с использованием различных материалов (научно-технических публикаций, материалов конференций и выставок, каталогов фондов программ, рекламных проспектов) было проанализировано около 200 отечественных инструментальных средств, представляющих в основном разработки последних 5-7 лет. При этом инструментальные средства для анализа отбирались по двум критериям: соответствия функционального назначения анализируемого средства проблематике решения всего множества задач, связанных с проектированием, разработкой и сопровождением прикладного ПО; существования завершенной программной реализации анализируемого средства и наличия сведений о его тестировании либо об опыте практического использования. Отметим, что рыночные критерии в данном анализе не участвовали. Далее приведен ряд интегральных характеристик, полученных по исследованной совокупности отечественных инструментальных средств.

**Функциональная ориентация инструментальных средств.** Распределение количества отечественных инструментальных средств в пределах анализируемой совокупности по их функциональному назначению имеет следующий вид:

– выполнение автономных задач и частных операций — 24 средства;
– поддержка начальных стадий создания программного проекта — 6;
– реализация разработки и производства программного продукта — 79;
– обеспечение работ по управлению проектированием — 10;
– поддержка полного жизненного цикла ПО или его значительной части — 29;
– поддержка создания узко ориентированного прикладного ПО — 29.

**Аппаратная платформа работы инструментальных средств.** Распределение отечественных инструментальных средств по типам ЭВМ показано на рис. 1. Общее количество типов ЭВМ, которые были отмечены при исследовании инструментальных средств, составило примерно 25; наиболее распространены средние модели ЕС ЭВМ, СМ ЭВМ, а также IBM PC. Возможности унификации процесса разработки ПО путем использования одного и того же инструментального средства на разных аппаратных платформах характеризуется следующим соотношением: 65% средств ориентировано на один тип ЭВМ, 35% на два и более. Для зарубежных CASE средств это соотношение выглядит как 58% и 42%, то есть за рубежом степень унификации несколько выше, чем у нас.

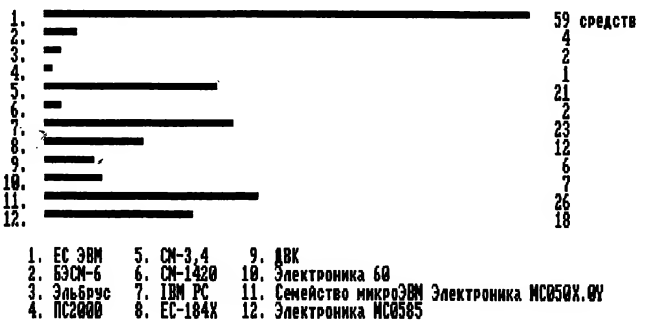



**Рис. 1. Аппаратная платформа работы инструментальных средств**

**Языки программирования.** Общее число языков с учетом их диалектов составило около 70. В число трех наиболее часто применяемых (рис. 2) входят автокоды специализированных и встроенных ЭВМ, языки группы Fortran и PL/1. Близко к ним идут C, Pascal, Assembler EC. В отличие от зарубежных CASE средств, где ассемблеры и автокоды практически отсутствуют, в отечественных средствах они составляют наибольший процент (27%). Один язык программирования поддерживает 55% средств, два и больше — около 20% (13% средств не поддерживает ни одного языка; по 12% нет данных). При этом, если поддерживается два языка, в связке идут, как правило, язык высокого уровня и ассемблер. Для зарубежных CASE средств соотношение систем, поддерживающих один язык и более одного языка, составляет 41% и 30%; как и раньше, это означает меньшую степень унифицированности наших средств.

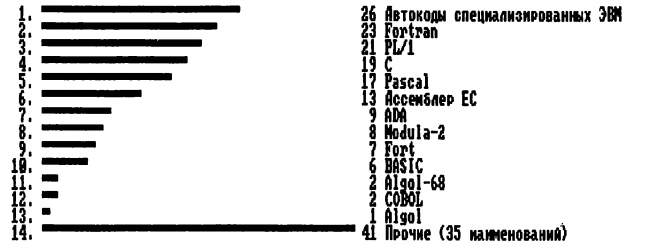



**Рис. 2. Языки программирования**

## Анализ отечественных инструментальных средств

К числу распространяемых на коммерческой основе отечественных инструментальных средств последних двух-трех лет разработки, как показывают рекламные публикации и материалы выставок Софтул и Информатика, могут быть отнесены следующие инструментальные средства (они перечислены по уровням интеграции).

**Автономные инструменты:**

ТехноСистем (НПП «ТЕХНОСИСТЕМ», г. Воронеж);

Комплекс инструментального ПО «Овсень» (промышленное товарищество «Овсень»);

Infosys (МП «БИТ», г. Санкт-Петербург);

Диаграммер (Рос НИИ ИТ и АП);

CASE.Аналитик (фирма «Эйтекс»);

Комплекс сертификационных испытаний CASTE (СНПО «Алгоритм»).

**Системы программирования 4GL:**

Система CASE-RT (Рос НИИ ИТ и АП).

**Интегрированные CASE средства:**

СИНТЕЗ (Академия им.Дзержинского).

**Интегрированные пакеты:**

МАСТЕР (СП «Микроинформ»);

ИРИС (фирма «Инфорг»).

Характеристики перечисленных инструментальных средств приведены в табл. 1. Более подробно с некоторыми из этих средств можно ознакомиться в [3]-[6], с другими — путем консультаций с разработчиками и поставщиками. Ограниченность приведенного списка во многом отражает то обстоятельство, что большое число известных разработчиков технологий и инструментальных средств (как правило, в крупных государственных фирмах, создающих сложное прикладное ПО) по ряду причин не рассматривает свои технологии и инструменты как коммерческий продукт и ограничивает их распространение и использование внутренними разработками фирмы.

Предварительный анализ, проведенный на основе этих характеристик, показал, что главные недостатки отечественных инструментальных средств — это преобладание автономных инструментальных средств и крайне ограниченное количество мощных интегрированных CASE средств; несоответствие требованиям, которые предъявляются современными прикладными программами; слабая унификация различных инструментальных CASE средств; перемещение центра тяжести разработок инструментальных средств на платформу IBM PC и, соответственно, сокращение разработок на больших ЭВМ. В результате как разработчиков прикладных программ, так и конечных пользователей весьма слабо привлекают имеющиеся в обращении инструментальные средства (за исключением узкого круга инструментов), и они либо ориентируются на зарубежные инструменты, либо вынуждены (в том числе из-за недостатка валютных средств) создавать их самостоятельно для собственного применения. Анализ также показывает, что нынешнее состояние дел в области разработки инструментальных CASE средств не дает пока оснований для скорого кардинального улучшения дел, поскольку продуктов, которые полностью отвечали бы современному уровню требований, были бы рассчитаны на длительную перспективу и составили бы конкуренцию зарубежным CASE средствам, в отечественном парке инструментальных средств практически нет.

Можно выделить целый ряд причин, определяющих неудовлетворительное состояние дел в этой области. Однако в большинстве случаев они связаны со сложившейся в настоящее время ситуацией в области вычислительной техники и компьютерной технологии, для которой характерны следующие особенности:

- широкое использование персональных ЭВМ, где пока еще невозможна реализация большого и сложного ПО и потому хватает штатных систем программирования;
- массовое пиратское и неконтролируемое распространение разнообразного программного обеспечения, в том числе и самой технологии с программными инструментальными средствами;
- крайняя неэффективность экономической и организационной инфраструктуры распространения и внедрения ПО, из-за чего инструментальные средства, в особенности имеющие ограниченные функции, многие пользователи предпочитают изготавливать и устанавливать самостоятельно;
- отсутствие маркетинговых исследований в области изучения потребностей рынка программных продуктов.

Анализ практической деятельности по разработке прикладного ПО показывает, что в сложившейся ситуации многие пользователи персональных, а нередко и больших ЭВМ обходятся при создании своих программ ограниченными инструментальными средствами, входящими в штатные ОС, и в первую очередь окружением языка программирования. При таком подходе достаточно типичная организация работ по созданию ПО заключается в том, что берутся имеющиеся в системном ПО компиляторы, редакторы, отладчики, утилиты и другие подобные средства, объединяются в некоторый пакет, иногда вводятся дополнительные инструменты, в том числе и собственной разработки, и вся эта совокупность средств применяется в качестве инструментальной поддержки принятой в данном коллективе технологии создания ПО.

Серьезным препятствием на пути унификации и массового распространения инструментальных средств является отсутствие соответствующих стандартов. По мнению зарубежных специалистов, именно стандарты сегодня являются тем самым недостающим звеном в программировании, отсутствие которого сегодня стало тормозом на пути его становления и стабилизации. Поэтому в настоящее время большое значение придается концепции открытых систем в информационных технологиях вообще и в области инструментальных средств в частности.

## Направления и перспективы развития инструментальных средств

К числу наиболее важных отечественных тенденций, которые определяют дальнейшее развитие и ис-

Таблица 1

Характеристики отечественных инструментальных средств

| Название средства, фирмы-поставщика | Назначение инструментального средства | Состав средства | Языки программирования | Аппаратная платформа |
|---|---|---|---|---|
| **Автономные инструментальные CASE средства** | | | | |
| Техносистем<br><br>«ТЕХНО-СИСТЕМ», г.Воронеж | Методология основана на трех положениях: функционально-структурных схемах; блочно-иерархическом принципе; графическом образе. Основные этапы разработки программных продуктов: формализация задачи в виде графа; отображение графа на экране и конструирование оконного интерфейса; информационное описание данных; интерпретация и трансляция графа на языки Pascal, C, Clipper. | — | Pascal, C, Clipper | IBM PC |
| Infosys<br><br>МП «БИТ» С.-Петербург | Предназначена для создания и сопровождения информационных систем; обеспечивает описание и создание файлов БД, экранных форм и отчетов; подключение нестандартных функций; разработку новых приложений на основе ранее созданных; организацию диалога типа меню и управление им; поиск, просмотр, добавление, изменение и удаление информации из файлов БД с простой и иерархической структурой; просмотр записей отчетов прикладных систем, их фильтрацию, сортировку по любым полям и сочетаниям. | Содержит файлы словарей-справочников, генератор прикладных систем и исполнительное ядро прикладных систем. | — | IBM PC |
| Диаграммер<br><br>РосНИИ ИТ и АП | Программный продукт для поддержки процесса создания информационных систем, позволяющий описывать структуру и функции системы в наглядной и удобной графической форме. Реализует четыре вида диаграмм: «сущность-связь», потоки данных, диаграмму функций, матричную диаграмму. | — | — | IBM PC |
| Комплекс инструментального ПО «Овсень»<br><br>ПТ «Овсень» | Комплекс инструментального ПО «Овсень» предназначен для создания средств автоматизации инженерного труда, систем сбора и обработки информации, контрольных систем, систем управления движением, систем охранной сигнализации. Средства ориентированы на взаимодействие с базой данных db_VISTA. | Многозадачный объектно-ориентированный диспетчер реального времени, экранный сервер, генератор интерфейсов, генератор отчетов, встраиваемые электронные таблицы. | C/C++ | IBM PC |
| CASE.Аналитик<br><br>Фирма «Эйтекс» | Инструментальное средство системного аналитика. Включает следующие функции: структурный анализ информационных систем и нотации Gane/Sarson; построение иерархической информационно-логической модели системы; графический интерфейс Windows; визуальное проектирование: диаграммы потоков данных; структурные диаграммы потоков, миниспецификации процессов; база данных проекта системы в формате Paradox; документирование и средства презентации проекта; верификация проекта. | — | — | IBM PC |
| Комплекс CASTE<br><br>СНПО «Алгоритм» | Инструментально-методический комплекс сертификационных испытаний программных средств. Ввключает три этапа: измерение характеристик программного продукта, оценка его качества и принятие решения о соответствии оценок требованиям к качеству продукта. В рамках каждого этапа может быть построена система показателей качества для конкретного продукта; проведены модельные испытания программного средства; сформировано заключение о его качестве; выдан сертификат соответствия продукта предъявляемым требованиям; рассчитаны риски, связанные с использованием этого продукта. | — | — | ЕС ЭВМ, IBM PC |

пользование инструментальных CASE средств, следует отнести усиливающийся интерес и растущее понимание той определяющей роли технологий и соответствующих средств автоматизации, которую они играют в создании эффективного прикладного ПО. Следует ожидать, что эта тенденция окажет стимулирующее воздействие на развитие и расширение отечественных работ в данной области. Необходимо всемерно поддерживать эту тенденцию, поскольку, несмотря на растущую роль международного разделения труда, кооперации и специализации, существуют такие ключевые сферы, где полная зависимость

Таблица 1 (продолжение)

Характеристики отечественных инструментальных средств

| Название средства, фирмы-поставщика | Назначение инструментального средства | Состав средства | Языки программирования | Аппаратная платформа |
|---|---|---|---|---|
| **Системы программирования 4GL** | | | | |
| CASE-RT РосНИИ ИТ и АП | Обеспечивает поддержку разработки многофункциональных программных систем, не относящихся к классу информационных: системы обработки данных измерительных приборов, моделирования процессов, гипертекстовых справочных систем, обучающих подсистем. Поддерживаются функции прототипирования программной логики и интерфейса, моделирования и др. Библиотеки программ содержат функции управления программной системой, средства работы с интерфейсом и функции обработки данных и формирования отчетов. | Включает технологический комплекс для интерактивного проектирования элементов прикладной системы и пользовательского интерфейса, библиотеки программ, включаемые в состав пользовательского задания. | C | IBM PC |
| **Интегрированные инструментальные CASE средства** | | | | |
| СИНТЕЗ<br><br>Академия им. Дзержинского | Инструментальные средства разработки информационных систем. Основаны на СИНТЕЗ методологии, которая определяет состав и последовательность этапов разработки, организацию взаимодействия исполнителей, постановки проектных работ на разных этапах жизненного цикла, методы и алгоритмы их решения, критерии и процедуры оценки качества проектных решений, формы и средства спецификации исходных данных проектных решений и материалов. Основные особенности: системный подход к проектированию, охват всех этапов жизненного цикла, концептуальное проектирование СУБД и оптимизация концептуальных проектов по многим критериям. | ПРОРОК — автоматизированная система концептуального проектирования баз данных.<br><br>СИНТЕЗ-Т — инструментальная поддержка процессов администрирования, проектирования и разработки средних и больших информационных систем.<br><br>СИНТЕЗ+ - базовые компоненты для комплектования АРМ разработчиков всех основных категорий. | — | ЕС ЭВМ, IBM PC |
| **Интегрированные пакеты** | | | | |
| МАСТЕР<br><br>СП «Микроинформ» | Интегрированная система МАСТЕР 1.0 - многокомпонентное инструментальное средство для разработки и внедрения современных информационных технологий. Сфера применения: разработка прикладных систем в области автоматизации управленческой, научной и производственной деятельности (бухгалтерия, САПР, кадры, обучающие программы, гостиничные, медицинские и складские системы, банковские проекты и т.д.). Отличительные особенности: ориентация на отечественных пользователей; решение повседневных задач конечными пользователями и разработка прикладного обеспечения программистами; русскоязычный интерфейс; полная поддержка русскоязычных данных; греческие, математические и химические символы на экране и принтере. | Полная аппаратно-независимая поддержка русского языка; текстовый редактор; электронные таблицы; средства деловой и иллюстративной графики; система управления базами данных; язык программирования МАСТЕР; единая визуальная среда для структурирования, обработки и хранения информации и программ; средства работы в локальных сетях; экспорт/импорт данных в стандартных форматах (ASCII, DIF, dBASE, ЛЕКСИКОН). | МАСТЕР | IBM PC |
| ИРИС 2.0 «Инфорг» | Набор инструментальных средств для организации обработки деловой информации на современном уровне. Идеология базируется на отказе от традиционной реляционной модели данных. Это обеспечивает большую скорость и эффективность работы по сравнению с известными СУБД. Наиболее эффективные сферы использования: создание прикладных систем в офисной деятельности (кадры, бухгалтерия, планирование и контроль работ), гостиничные, медицинские, складские системы, автоматизация муниципальной деятельности. | — | — | IBM PC |

от партнеров является неоправданной. Подтверждением того, что технологии и автоматизированные средства создания ПО относятся именно к такой сфере, является наличие до недавнего времени инструментальных средств и технологий в списках COCOM.

Исходя из реально существующей в области информационных технологий ситуации, наиболее вероятно ожидать, что дальнейшее развитие технологий и инструментальных средств пойдет в двух различных направлениях, определяемых аппаратной платформой.

В области персональных компьютеров — это **массовая разработка разнообразных инструментальных программных средств**, охватывающих относительно ограниченную совокупность технологических функ-

ций, сводящихся в основном к наборам утилит, системных программ или штатных средств ОС. Эти инструменты, как правило, разрабатываются в собственных интересах, слабо приспособлены к интеграции и мало унифицированы. Они охватывают ПО из так называемых некритических областей, которое и составляет основную часть ПО на этой платформе. Примером типового современного инструментального средства для программиста работающего на персональных компьютерах, является созданный фирмой Borland пакет системных программ Turbo C. Он включает отладчик, ассемблер и модуль для определения времени выполнения участков программ. Никаких дополнительных средств программирования не требуется, так как ассемблер может быть объединен с компилятором языка C++. Отладчик позволяет создать несколько окон на экране, а также вести отладку на удаленном ПК, связанном с основным ПК линией связи.

**Второе направление** — это область больших ЭВМ, призванная реализовывать ПО систем реального времени и управляющих систем. Наряду с большими ЭВМ здесь часто фигурируют и специализированные или встроенные ЭВМ, которые ориентированы на решение определенных задач и очень экономно расходуют свои ресурсы, из-за чего системные и тем более инструментальные средства обеспечения работ у них развиты слабо. В большинстве случаев здесь широко распространена коллективная работа больших групп специалистов, что требует соответствующих подходов и средств. Поэтому в этой области имеются хорошие предпосылки для формирования развитой технологии и мощных инструментальных систем, обеспечивающих высокую степень интеграции, унификации, поддержанной возможностью адаптации инструментов, и, как следствие, достаточно широкую тиражируемость этих средств.

В настоящее время появилось третье направление, связанное с рабочими станциями. Оно занимает промежуточное положение между персональной техникой и большими ЭВМ и на Западе рассматривается как одна из основных перспектив развития компьютерной техники. Рабочие станции ориентируются в основном на UNIX или UNIX подобные ОС, и это является одним из принципиальных требований к инструментальным средствам разработки прикладного ПО для данного класса техники. В силу ряда причин у нас в стране это направление пока не относится к распространенному, однако, судя по существующим тенденциям, оно имеет хорошие перспективы и является многообещающим.

С точки зрения научно-технических перспектив развития технологий и инструментальных средств создания ПО, в качестве наиболее вероятных можно выделить следующие пути и решения.

**Повторная разработка ПО.** В основе лежит принцип итерационного подхода к получению готового программного продукта путем многократного использования или модернизации исходного решения, проекта, программы, группы программ и так далее в рамках так называемой спиральной модели жизненного ПО.

**Объектно-ориентированное программирование**, как и повторная разработка, направлено на повышение эффективности разработки и экономических показателей ПО.

**Метапрограммирование** предполагает, что программирование производится на высоком уровне абстракции с использованием спецификаций; для получения же традиционного исходного кода (который может быть преобразован в машинный код существующими средствами) необходима разработка специального транслятора.

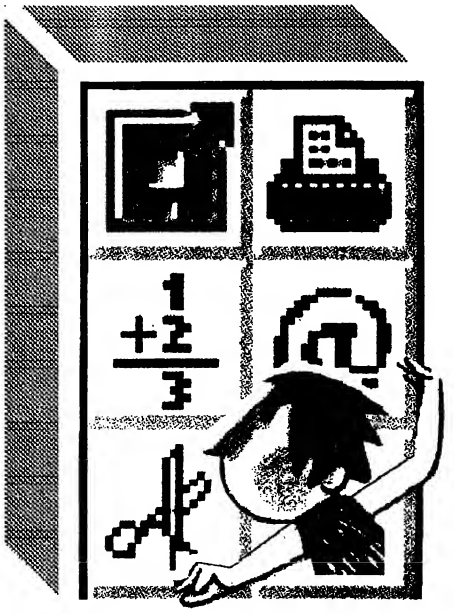
**Искусственный интеллект для создания ПО** включает три области исследования: автоматическое программирование, где основная задача состоит в формализации модели и техники построения процесса программирования; построение средств управления жизненным циклом функционирования проектируемой системы, учитывающих все требования, аспекты, факторы и так далее, возникающие в процессе создания ПО; системы управления пользовательским интерфейсом со встроенным интеллектом для быстрого макетирования средств отображения пользовательского интерфейса, поскольку именно интерфейс становится определяющим фактором эффективности создаваемых систем.

В ряду этих перспективных направлений стоит и концепция открытых систем в области информационных технологий и инструментальных средств. ■


Литература

1. А. Штрик. CASE: автоматизированное проектирование программного обеспечения. Концепции и основные положения// Монитор.– 1992.– № 4.
2. А. Штрик. Зарубежные CASE средства создания программного обеспечения в отечественных разработках// Монитор.– 1993.– № 3.
3. Викторова Н.П., Паронджанов С.Д. Особенности проектирования систем реального времени и требования к CASE технологии разработки таких систем/ Семинар «CASE технологии»: Сб. материалов, Москва, 1992.
4. Ветошкин В.М., Гузенко В.Г. «Синтез» — методология, технология и семейство инструментальных средств разработки информационных систем/ Семинар «CASE технологии»: Сб. материалов, Москва, 1992.
5. Веселов Е.М. Интегрированная система «МАСТЕР» для ПЭВМ.– М.: Финансы и статистика, 1990.
6. Чижов С.С. Комплекс инструментально-методических средств оценки качества и сертификации ПС CASTE: основные принципы/ Всесоюзная конференция «Качество программного обеспечения»: Сб. материалов.– М.: СНПО «Алгоритм», 1991.

# Программируем СУБД на Turbo Vision



*С.Никоненко*

## Часть 2

*Алиса уже не раз хотела встать и сказать: «Большое спасибо за Ваш интересный рассказ», — но она все-таки немножко надеялась, что продолжение следует, и крепилась: сидела смирно и молчала.*

*Льюис Кэрролл.*
*Приключения Алисы в стране чудес*

### Обзор объектов Data Vision

Библиотека Data Vision интегрирует базы данных (БД) в систему объектов Turbo Vision. По этой причине все объекты Data Vision наследуют объектам Turbo Vision. Иерархия этого наследования показана на рис. 1. Объекты Turbo Vision обведены одинарной рамкой, а наследники из Data Vision — двойной. По связям наследования и далее по тексту Вы увидите, что объекты Data Vision можно объединить в 3 различные группы:

- объекты баз данных;
- объекты видимых элементов;
- вспомогательные объекты.

Рисунок 1 не содержит некоторых второстепенных объектов Data Vision (вспомогательных объектов администратора БД приложения, процентовок, аналогов календаря и калькулятора Turbo Vision).

Несмотря на большое искушение сразу перейти к видимым элементам и продемонстрировать мощь и элегантность специально разработанных полноэкранных редакторов баз данных, вероятно, лучше предварительно детально рассмотреть основы работы с физическими базами данных, а затем использовать контекст видимых элементов для демонстрации готовых приложений. В качестве рабочего приложения будем использовать интерактивное расширение первого примера — утилиту SIMPLE, которая организует многооконный интерфейс с файлами в форматах dBase III+ и Paradox v.3.5 одновременно с простым взаимным преобразованием файлов.

В Data Vision любая БД является объектом. Свойства объекта заключаются в его методах. Общие свойства объектов БД обеспечиваются их наследованием единому предку TDefDB (рис. 1). Этот объект имеет набор виртуальных методов, которые, собственно, и являются унифицированным интерфейсом с любым типом реляционных (либо приведенных к реляционным) БД. Те из вас, кто уже знаком с полиморфизмом и наследованием, не нуждаются в комментариях этого положения. Для остальных поясню: каждый из объектов-наследников TDefDB перекрывает его виртуальные методы и таким образом обеспечивает определенный стандарт программного интерфейса с объектом. Например, для удаления записи объект TDefDB и любой его наследник подразумевает использование метода TEDelRec. Вы, конечно, можете использовать другой метод, назвать его иначе, даже сделать его статическим, но в этом случае Вы лишаете себя преимуществ объектно-ориентированного подхода Data Vision. Допустим, Ваш объект называется MyDBObject, а метод удаления записи в файле БД носит название DeleteRecord. Вот что у Вас получится при написании процедуры удаления записи:

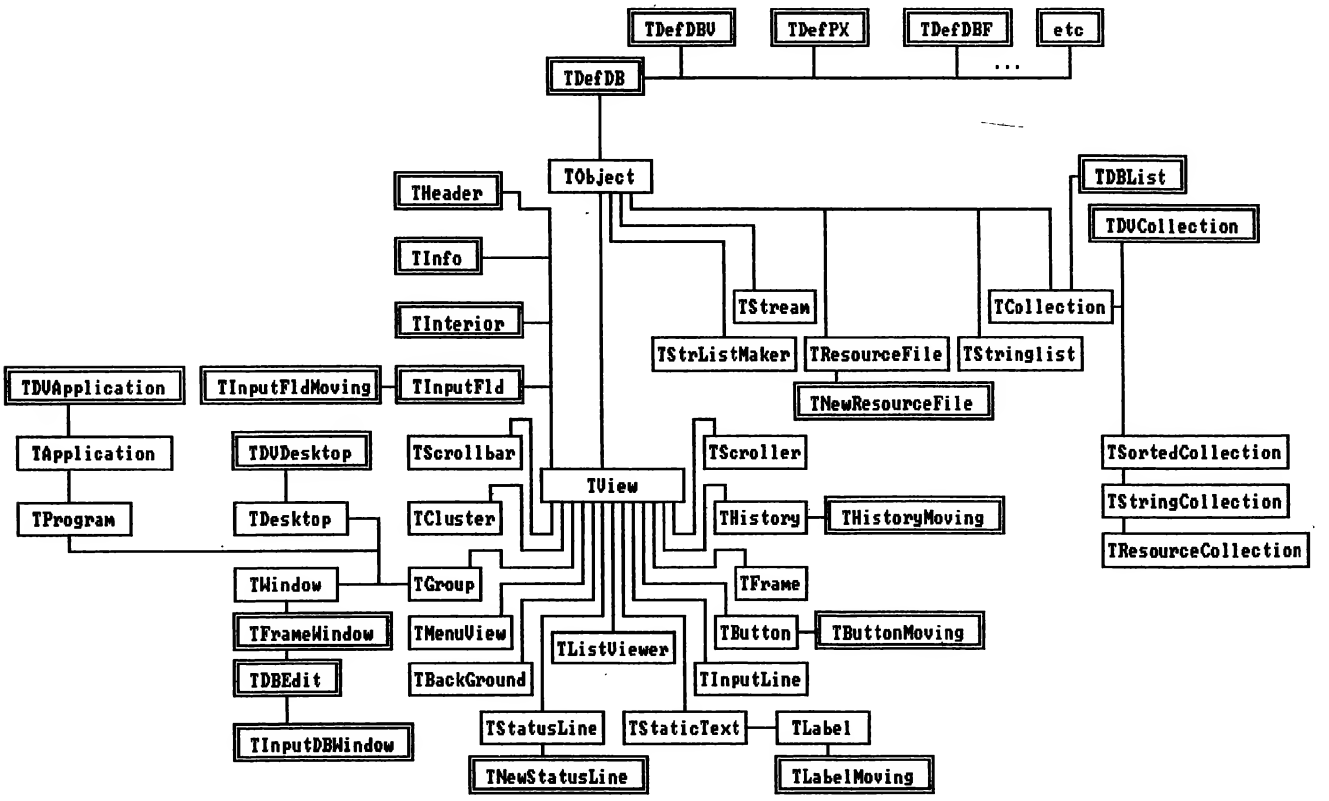



**Рис. 1**

```
(* Процедуре передается указатель
   на экземпляр объекта *)
procedure DeleteDBRecord (Item:
                          pMyDBObject);
begin
   Item^.DeleteRecord      { Удаление записи }
end;
```

Любой искушенный в ООП программист написал бы это иначе:

```
(* Процедуре передается указатель
   на экземпляр объекта *)
procedure DeleteDBRecord (Item: pDefDB);
begin
   Item^.TEDelRec          { Удаление записи }
end;
```

– В чем тут разница? — воскликнет новичок. — Эффект один и тот же!

Тот же, да не тот. Второй пример кода будет ОДИНАКОВО работать для экземпляров любых типов объектов БД, унаследованных от TDefDB, первый же только для Вашего. В этом и состоит смысл унифицированного интерфейса и один из аспектов ООП в целом: один и тот же набор виртуальных методов может использоваться для всей иерархии наследников объекта. Вообще говоря, технология ООП всегда «заставляет» программиста унифицировать свои представления о предметной области, и те, кто давно занимается объектно-ориентированным подходом, видимо, уже и не представляют себе, как может быть иначе.

TDefDB имеет следующие группы методов:

1) информационные;
2) навигационные;
3) методы доступа к данным;
4) сетевую функцию;
5) пользовательскую функцию;
6) вспомогательные.

Поскольку статья, помимо обзорного, имеет также справочный характер, в таблице 1 перечислены в алфавитном порядке виртуальные методы унифицированного интерфейса с объектами БД с указанием параметров, краткими аннотациями, ссылками на принадлежность к определенной группе методов и некоторой дополнительной информацией. Если в качестве этой информации указано слово «абстрактный», это означает, что метод является абстрактным, то есть его тело состоит из вызова процедуры Abstract Turbo Pascal, и обращение к нему напрямую вызовет появление 211 ошибки во время выполнения. Если здесь же указано слово «псевдоабстрактный», это означает, что метод просто-напросто пустой.

Помимо описанных в таблице виртуальных методов, существует также ряд активно используемых статических методов доступа к данным: TEGetXX и TEPutXX, которые читают содержимое поля БД и

Таблица 1

**Виртуальные методы унифицированного интерфейса Data Vision**

| Описатель метода | Аннотация | Метод | Дополнительная информация |
|---|---|---|---|
| procedure TEAddRec(N: Integer); | Добавляет в конец БД N записей | Вспомогательный | Абстрактный |
| function TEBOF: Boolean; | Возвращает True, если установлено состояние "Верх БД" | Информационный | Абстрактный |
| function TEColor: Byte; | Возвращает цвет окраски текущей записи в редакторе таблиц | Вспомогательный | |
| procedure TECommit; | Функция сброса буфера БД | Вспомогательный | Абстрактный |
| procedure TECopyRec; | Копирует текущую запись в новую последнюю запись | Вспомогательный | Абстрактный |
| procedure TEDBChanged; | Сообщает приложению об изменении объекта БД | Вспомогательный | |
| procedure TEDBColorChanged; | Сообщает приложению о необходимости перерисовать видимые элементы, связанные с объектом БД | Вспомогательный | |
| function TEDBName: String; | Возвращает имя БД | Информационный | |
| function TEDBType: String; | Возвращает тип БД | Информационный | |
| procedure TEDelRec; | Удаляет текущую запись | Вспомогательный | Абстрактный |
| function TEEOF: Boolean; | Возвращает True, если установлено состояние "Конец БД" | Информационный | Абстрактный |
| function TEEmpty(N: Byte): Boolean; | Возвращает True, если поле N является "пустым" для данного формата поля | Информационный | Абстрактный |
| function TEFldName(N: Byte): String; | Возвращает имя N-го поля | Информационный | |
| function TEFldNo(AName: String): Byte; | Возвращает номер поля по имени AName | Информационный | |
| procedure TEGetField(N: Byte; FType: Char; Var AVal); | Читает из N-го поля в буфер данных AVal по формату FType | Доступа к данным | Абстрактный |
| procedure TEGetFldForm(N: Byte; Var FType: Char; Var Len,Dec: Byte); | Возвращает информацию о формате N-го поля: тип FType, ширину Len, количество десятичных разрядов Dec | Информационный | Абстрактный |
| function TEGetLen(N: Byte): Byte; | Возвращает длину N-го поля | Информационный | Абстрактный |
| procedure TEGoBottom; | Перемещение к последней записи БД | Навигационный | Абстрактный |
| procedure TEGoNext; | Перемещение к следующей записи. Если БД открыта с индексом, то перемещение по индексу | Навигационный | Абстрактный |
| procedure TEGoPrev; | Перемещение к предыдущей записи. Если БД открыта с индексом, то перемещение по индексу | Навигационный | Абстрактный |
| procedure TEGoto(N: LongInt); | Перемещение к N-й записи | Навигационный | Абстрактный |
| procedure TEGoTop; | Перемещение к 1-й записи | Навигационный | Абстрактный |
| function TEHint(AWhat: Byte):Word; | Возвращает контекст подсказки при загрузке/сохранении БД и в других случаях | Вспомогательный | |
| function TEIndexName: String; | Возвращает имя индекса | Информационный | |
| procedure TEInsertRec; | Добавляет запись в текущую позицию БД | Вспомогательный | |
| function TEIsMarked: Word; | Возвращает тип отметки текущей записи | Информационный | |
| function TEIsMyDB(ADBMS: pDefDB): Boolean; | Возвращает True, если ADBMS является данным объектом БД | Информационный | |
| function TELocate(N: Byte; FType: Char; Var AVal): Boolean; | Осуществляет поиск по N полю. Ключ поиска в буфере поиска N представлен в формате FType | Вспомогательный | Абстрактный |
| procedure TEMark(AType: Word; Enable: Boolean); | В зависимости от Enable устанавливает (Enable=True) или снимает (Enable=False) тип отметки AType | Вспомогательный | Псевдоабстрактный |
| procedure TENetWork(AOperation: Word; Var AVal); | Выполняет сетевую операцию AOperation | Сетевая функция | |
| function TEOrderNo: LongInt; | Возвращает номер текущей записи по индексу | Информационный | |
| function TERecNo: LongInt; | Возвращает номер текущей записи | Информационный | Абстрактный |
| procedure TEReplace(N: Byte; FType: Char; Var AVal); | Записывает в N-е поле данные из буфера AVal, представленные в формате FType | Доступа к данным | Абстрактный |

Таблица 1 (окончание)

| | | | |
|---|---|---|---|
| procedure TERestorePos(N:LongInt); | Метод, обратный TESavePos, восстанавливает запись под номером N как текущую | Вспомогательный | |
| procedure TESavePos(Var N:LongInt); | Сохраняет в N номер текущей записи аналогично TERecNo | Вспомогательный | |
| procedure TESaveRec; | Сохраняет в БД буфер текущей записи | Вспомогательный | Псевдоабстрактный |
| function TETotalFld: Byte; | Возвращает общее количество полей заголовка БД | Информационный | Абстрактный |
| function TETotalMarks(AType: Word): LongInt; | Возвращает количество записей, отмеченных типом AType | Информационный | |
| function TETotalRec: LongInt; | Возвращает общее количество записей БД | Информационный | Абстрактный |
| procedure TEUser(Var Event: TEvent; ADBEdit: PFrameWindow; TEMode: Byte); | Отрабатывает события Event, направляемые специальными видимыми элементами ADBEdit | Пользовательская функция | Псевдоабстрактный |
| procedure TEZap; | Удаляет все записи БД | Вспомогательный | Абстрактный |

записывают информацию в него по определенному формату. Эти методы приведены в таблице 2. Формат, используемый методом, соответствует формату параметра метода при записи либо формату возвращаемого при чтении значения.

Таблица 2

**Методы форматного чтения/записи данных в Data Vision**

| Описатель метода | Чтение/запись | Формат |
|---|---|---|
| procedure TEGetDate(N: Byte; Var AYY,AMM,ADD: Word); | Чтение | Дата |
| function TEGetInt (N: Byte): Integer; | Чтение | Короткое целое |
| function TEGetLong(N: Byte): LongInt; | Чтение | Длинное целое |
| function TEGetReal(N: Byte): Real; | Чтение | Вещественное |
| function TEGetStr (N: Byte): String; | Чтение | Строка |
| procedure TEPutDate(N: Byte; AYY,AMM,ADD: Word); | Запись | Дата |
| procedure TEPutInt (N: Byte; AInt: Integer); | Запись | Короткое целое |
| procedure TEPutLong(N: Byte; ALong: LongInt); | Запись | Длинное целое |
| procedure TEPutReal(N: Byte; AReal: Real); | Запись | Вещественное |
| procedure TEPutStr (N: Byte; AStr: String); | Запись | Строка |

Кроме всех описанных методов, существуют деструктор и конструкторы, метод сохранения в потоке и некоторые другие, не приведенные в таблицах.

Использование методов в примерах (в первую очередь не описанных) будет активно комментироваться.

Может возникнуть резонный вопрос о совместимости форматов полей различных типов БД. Этот вопрос также является компонентом унифицированного интерфейса с объектами БД. На этот счет существуют определенные соглашения, о которых будет рассказано после рассмотрения разновидностей объектов БД.

Все используемые объекты БД порождены, как уже утверждалось, типом TDefDB. Data Vision использует следующие разновидности объектов БД:

- объекты физических БД;
- объект виртуальной БД;
- унаследованные объекты.

Ядро Data Vision — объект виртуальной базы данных — требует отдельного внимания, и мы к нему еще вернемся позже. До сих пор мы имеем дело только с объектами физических БД. Они основаны на физическом интерфейсе низкого уровня с конкретным типом БД. В примерах 1 и 2 используются экземпляры объектов, работающих с базами данных форматов dBase и Paradox.

Объекты, унаследованные от объектов физических БД или объекта виртуальной БД, так и называются унаследованными.

При чтении примера 1 Вы, вероятно, уже спрашивали себя, каким образом сопрягаются форматы полей dBase и Paradox; ведь dBase не имеет форматов «short» и «currency», а Paradox v3.5 не располагает ни логическими, ни Memo полями, а кроме того, не имеет понятия о «знаках после точки», известных всем, кто работал с dbf форматом. Здесь действуют соглашения двух уровней: создания заголовка файла БД и доступа к полям БД.

При создании заголовка БД соответствующая функция типа XXCREATE (DBFCreate в примере 1; она же ParaCreate в примере 2) обязана интерпретировать известные символы формата соответствующим корректным образом, например, задание «денежного» формата «currency» при создании dbf файла интерпретируется как числовой формат N. При создании Вашего собственного интерфейса Вы обязаны помимо собственных форматов корректно отработать и символы форматов dBase и Paradox. Благодаря этому функция CreateFromDB из примера 1 работает корректно при формировании заголовка dbf файла из Paradox файла.

Я думаю, что с созданием заголовка файла БД все более или менее ясно, хотя остается еще метод TEGetFldForm, который в Paradox объекте возвращает для всех типов числовых полей некоторые зна-

чения количества знаков после запятой и общей ширины поля в знаках по умолчанию. Однако это не имеет прямого отношения к рассматриваемому аспекту унифицированного интерфейса: какие типы данных реально используются при работе с объектами баз данных?

Ответ напрашивается сам собой — это типы данных Turbo Pascal. Для работы с датой в Data Vision введен тип записи TDateRec, состоящий из величин дня, месяца и года. Главным здесь является следующее: при чтении данных по какому-либо формату из поля БД метод TEGetField обеспечивает представление информации в необходимом формате вне зависимости от реального формата поля; при записи данных в поле БД параметр формата указывает на формат данных в передаваемом буфере, и метод TEReplace в обязательном порядке приводит эти данные к формату поля.

Разумеется, преобразование должно быть осмысленным; если Вы пытаетесь считать из поля в формате даты информацию по формату вещественного числа, Вы ее получите, но вряд ли она Вас устроит. Именно поэтому передача данных из файла Paradox в файл dBase (а также между файлами ЛЮБЫХ форматов) выглядит так просто:

```
(* Процедура копирования поля
   под номером N из текущей записи
   объекта БД InBase в текущую запись
   объекта БД OutBase.
   InBase и OutBase —
      указатели на объекты *)
procedure CopyOneField (InBase,OutBase:
                        PDefDB; N: Byte);
begin
   OutBase^.TEPutStr
                (N,InBase^.TEGetStr(N))
end;
```

В таблице 3 приведен список используемых форматов и их краткое описание.

Таблица 3

Типы данных, поддерживаемые Data Vision

| Обозначение | Размер в байтах | Тип данных |
|---|---|---|
| C | 0..255 | String/строка (TPascal) |
| D | 8 | TDateRec/дата (Data Vision) |
| R | 6 | Real/вещественный (TPascal) |
| L | 4 | LongInt/длинный целый (TPascal) |
| I | 2 | Integer/короткий целый (TPascal) |

Вашему вниманию предлагается интерактивная версия примера 1 SIMPLE.PAS. ■

**Пример 2**

```
{ ------------------------------------------------
        Пример использования Data Vision
  Утилита преобразователя Paradox <-> dBase.
            TURBO PASCAL v6.0-7.0
                  ***
        CopyRight (C) 1993, Никоненко С.Б.
  Лаборатория программных средств SHIFT

              SIMPLE.PAS
------------------------------------------------- }
Uses   Dos,
  Memory,Objects,Drivers,Views,Menus,Dialogs,App,MsgBox,StdDlg,
  DVUtil,DVApp,DVDefine,DVObject,
  DVDefDB,DVDBEdit,DVDBase,DVPX;

Const
  cmOpen   = 1100;       { Команда открытия файла БД }
  cmSave   = 1101;       { Команда сохранения файла БД из текущего окна}
  cmSaveAs = 1102;       { Команда сохранения файла БД под другим
                           именем и, возможно, в другом формате }

  hcDBEdit          = 2000;  { Контекст подсказки для редактора }
  LoadDBHistory     = 100;   { ID списка истории при открытии файла }
  SaveDBHistory     = 101;   { ID списка истории при сохранении файла }

Type
  { Обратите внимание на то, что предком объекта-приложения
  является объект Data Vision TDVApplication. Это обязательное
  условие рабочего состояния программы. }
  pIDE = ^IDE;
  IDE = object (TDVApplication)
        constructor     Init;
        destructor      Done; virtual;
        procedure       HandleEvent (var Event: TEvent); virtual;
        procedure       InitMenuBar; virtual;
        procedure       InitStatusLine; virtual;
  end;

Var
  Appl: pIDE;
  FileName: String;
function CreateFromDB (DBItem: pDefDB):pDBHeader; forward;

(*** IDE ***)
constructor IDE.Init;
begin
  TDVApplication.Init;
  EnginePath:='';          { Оверлей PXENGINE.OVL в текущем каталоге }
  InitParadox(@ParaDefault)  { Инициализация исполняющей системы
PARADOX }
end;

destructor IDE.Done;
begin
  TDVApplication.Done;
  ExitParadox          { Выход из исполняющей системы PARADOX }
end;

procedure IDE.HandleEvent (var Event:TEvent);
Var
  R: TRect;

  { Функция проверяет наличие файла БД с именем AName
  в списке администратора БД приложения (АБДП).
  Если он есть в списке, значит, этот файл открыт
  и известен программе.
  Объект АБДП будет подробно рассмотрен позже. }

  Function IsNewDB (AName: String): PDefDB;
       Function FindName (Item: PDefDB): boolean; far;
       begin
            FindName:=UpperStr(Item^.TEDBName) = UpperStr(AName)
       end;
  begin
       if DBList<>nil then       { Проверка наличия объекта АБДП }
       IsNewDB:=DBList^.FirstThat (@FindName)
       else IsNewDB:=nil
  end;

  { Функция проверяет корректность выбранного файла и
  возвращает True, если расширение файла совпадает с
  расширением dBase или Paradox }

  function TrueExtention (E: ExtStr): Boolean;
  begin
       TrueExtention:=False;
       if (UpperStr(E)<>'.DBF')and(UpperStr(E)<>'.DB') then
```

```
        begin
            MessageBox('Incorrect file name',nil,mfError+mfYesButton);
            Exit
        end;
        TrueExtention: = True
end;

{ Функция открывает полноэкранный редактор для объекта БД NewDB
под идентификатором NewID }

procedure OpenDBEditor (NewDB: pDefDB; NewID: Byte);
Var
    WnCnt: Integer;          { Номер окна }
    DBE: pDBEdit;            { Указатель на экземпляр редактора таблиц }
begin
    NewDB^.TEGoTop;          { Установка БД на первую запись }
    WnCnt: = 0;              { Посылка сообщения Data Vision для
                             определения максимального номера окна }
    Message(DeskTop,evBroadCast,cmFillMaxWNum,@WnCnt);
    Inc(WnCnt);              { Получение первого "свободного" номера окна }
    with NewDB^ do
    begin
        R.Assign(1,1,20,20);     { Инициализация редактора Data Vision }
        DBE: = New(pDBEdit,Init(R,  { Границы видимого элемента }
            TEDBName,            { Заголовок окна (см. табл. 1) }
            WnCnt,               { Номер окна }
            '',                  { Имя фрейма (см. далее) }
            NewID,               { Уникальный ID объекта БД }
            False,               { Флаг указателя текущего элемента
                                 редактора }
            '',                  { Разделитель по умолчанию }
            MakeDBTitle(NewDB)   { Шапка таблицы }
        ));
        if (DBE = nil) then      { Переполнен пул надежности }
        begin
            MessageBox('Not enough memory for database editor',
            nil,mfYesButton+mfError);
            Exit
        end;
                                 { Окно может перестраиваться }
        DBE^.Options: = DBE^.Options or ofTileable;
        DBE^.HelpCtx: = hcDBEdit;
        with DeskTop^ do
        begin
        Lock;
        Insert(DBE);
        GetExtent(R);
        Cascade(R);              { Перестройка каскадом окон }
        UnLock
        end
    end
end;

{ Функция возвращает первый свободный идентификатор БД
из объекта АБДП }

function GetFreeID: Byte;
Var
NewID: Byte;
begin
    NewID: = 1;
    if DBList < > nil then
        while DBList^.IsInList(NewID) do Inc(NewID);
    GetFreeID: = NewID
end;

{ Процедура открывает объект БД и объект табличного
редактора для него }

procedure OpenDBase;
Var
W: pFileDialog;      { Указатель на диалоговое окно выбора файла }
Control: Word;       { Команда, указанная в диалоговом окне выбора файла}
D: DirStr;           { Путь файла }
N: NameStr;          { Имя файла }
E: ExtStr;           { Расширение файла }
NewDB: pDefDB;       { Объект БД }
NewID: Byte;         { Уникальный ID БД }
begin
    { Инициализация диалогового окна }
    W: = New(pFileDialog,Init('*.DBF','DataBase choice','DataBase ~n~ame',
        fdOkButton+fdHelpButton,LoadDBHistory));
    { Отработка диалогового окна с командой в Control }
    Control: = DeskTop^.ExecView(W);
    { Получение имени файла }
    W^.GetData(FileName);
    Dispose(W,Done);
    if Control < > cmFileOpen then Exit;
    FSplit(FileName,D,N,E);
    { Проверка на корректность типа выбранного файла }
    if not TrueExtention(E) then Exit;
    { Проверка на "известность" этого файла приложению }
    NewDB: = IsNewDB(FileName);
    if NewDB = nil then
    begin                        { Объект для этого файла не обнаружен }
        NewID: = GetFreeID;      { Получение уникального ID объекта БД }
        if UpperStr(E) = '.DB' then     { Инициализация объекта Paradox }
        NewDB: = New(pDefPX,Init(NewID,     { ID объекта }
            FileName,            { Имя файла }
            0,                   { Ключевое поле }
            False,               { Признак сохранения }
            0))                  { Количество полей в первичном индексе}
else                             { Инициализация объекта dBase }
        NewDB: = New(pDefDBF,Init(NewID,    { ID объекта }
            FileName,            { Имя файла }
            '',                  { Имя индекса }
            0,                   { Ключевое поле }
            0,                   { Признак уникального ключа }
            False))              { Признак автоиндексации }
    end
    else
    begin                        { Объект для этого файла обнаружен }
        DBInitError: = ieSuccess;       { Установка корректного значения
                                 глобальной переменной Data Vision,
                                 показывающей корректность инициализации
                                 объекта БД }
        NewID: = NewDB^.ID       { Получаем ID БД из объекта }
    end;
    if (DBInitError > ieExist) then
    begin                        { Если инициализация некорректна - выход }
        MessageBox('Can''t open database '+FileName,nil,
                mfOkButton + mfError);
        Exit
    end;
    NewDB: = DBP(NewID);  { Получение адреса объекта БД с ID = NewID.
                             Этот адрес необходим для объектов,
                             инициализируемых повторно. В этом
                             случае DBInitError = ieExist,
                             a NewDB = nil }
    OpenDBEditor(NewDB,NewID)    { Открытие окна редактора БД }
end;

{ Функция полного копирования информации из объекта БД
InBase в объект БД OutBase возвращает True, если
копирование прошло корректно }

function CopyBase (InBase, OutBase: pDefDB): Boolean;
Var
i: Byte;
begin
    CopyBase: = False;
    InBase^.TEGoTop; { Переход к верху БД }
    { Цикл, пока не конец входной БД и результат операции над БД
    корректен }
    while (not InBase^.TEEOF)and(InBase^.TEExitState = esSuccess) do
    begin
        OutBase^.TEAddRec(1); { Добавляем запись в выходную БД }
        for i: = 1 to InBase^.TETotalFld do  { Копируем запись }
        OutBase^.TEPutStr(i,InBase^.TEGetStr(i));
        OutBase^.TESaveRec;     { Сохраняем запись в выходной БД }
        InBase^.TEGoNext  { Переход к следующей записи входной БД }
    end;
    { Если скопировали до конца, копирование прошло нормально }
    if InBase^.TEEOF then CopyBase: = True
end;

{ Процедура сохранения БД }

procedure SaveDBase;
Var
W: pFileDialog;          { Диалоговое окно выбора имени БД }
Control: Word;           { Команда, выбранная в диалоговом окне }
TmpID, Io: Integer;
OutDB: pDefDB;           { Объект БД }
D: DirStr;               { Путь файла }
N: NameStr;              { Имя файла }
E: ExtStr;               { Расширение файла }
TmpStr: String;
F: File;
begin
        { Если нет ни одного окна - выход }
    if Desktop^.Current = nil then Exit;
        { Инициализация диалогового окна }
    W: = New(PFileDialog,Init('*.DBF','Save database as','Database ~n~ame',
        fdOkButton+fdHelpButton,SaveDBHistory));
        { Получение имени файла БД из текущего окна }
    TmpStr: = pDBEdit(Desktop^.Current)^.DBMS^.TEDBName;
        { Установка диалогового окна }
    W^.Filename^.SetData(TmpStr);
    Control: = DeskTop^.ExecView(W);
```

## Пример 2 (окончание)

```
        { Получение имени файла из диалогового окна }
    W^.GetData(FileName);
    Dispose(W,Done);
    if Control < > cmFileOpen then Exit;
    FSplit(FileName,D,N,E);
        { Проверка на корректность типа введенного файла }
    if not TrueExtention(E) then Exit;
        { Проверка наличия выходного файла }
    Assign(F,FileName);
    {$I-}
    Reset(F);
    Close(F);
    {$I+}
    Io: = Ioresult;  { Если файл не существует, либо существует, }
    if (Io < > 0) or  { но мы согласны его переписать }
        ((Io = 0) and
        (YesOrNo('This database already exist. Continue?',nil))) then
    begin
        OutDB: = IsNewDB(FileName);      { Проверка на известность этого
                                           файла приложению }
        if OutDB < > nil then            { Если выходная БД уже известна }
        begin                            { приложению }
        if OutDB = PDBEdit(Desktop^.Current)^.DBMS then
        begin      { Если это текущее окно, то просто сохраняем БД }
            Event.Command: = cmSave;
            PutEvent(Event);
            ClearEvent(Event);
            Exit
        end
        else
        if OutDB < > PDBEdit(Desktop^.Current)^.DBMS then
        begin                    { Если это не текущее окно }
            OutDB^.Free;  { Удаляем объект БД из приложения вместе
                            с его видимыми элементами }
            Erase(F);     { Удаляем выходной файл }
        end
        end
        else                             { Выходная БД неизвестна приложению }
        if Io = 0 then Erase(F);{ Если выходной файл есть, он удаляется }
        TmpID: = GetFreeID;{ Получение свободного уникального ID БД }
        if UpperStr(E) = '.DB' then
        begin                            { Если выходная БД в формате Paradox }
        ParaCreate(FileName,   { Создание заголовка }
        CreateFromDB(PDBEdit(Desktop^.Current)^.DBMS));
                        { Инициализация объекта физической БД }
        OutDB: = New(pDefPX,Init(TmpID,FileName,0,False,0))
        end
        else
        begin            { Если выходная БД в формате dBase }
        DBFCreate(FileName,    { Создание заголовка }
        CreateFromDB(PDBEdit(Desktop^.Current)^.DBMS));
                        { Инициализация объекта физической БД }
        OutDB: = New(pDefDBF,Init(TmpID,FileName,'',0,0,False))
        end;
        if (OutDB < > nil)and { Если копирование завешено нормально }
        (CopyBase(PDBEdit(Desktop^.Current)^.DBMS,OutDB)) then
        begin
            OutDB^.TECommit;             { Сброс БД на диск }
            OpenDBEditor(OutDB,TmpID) { Инициализация редактора БД }
        end
        else MessageBox('Can"t create database '+FileName,nil,mfError+
            mfYesButton)
    end
  end;

begin
  TDVApplication.HandleEvent (Event);
  If Event.What = evCommand then
  begin
      Desktop^.GetExtent(R);
      case Event.Commiand of
      cmOpen   : OpenDBase;        { Открытие БД для редактирования }
      cmSave   : if Desktop^.Current < > nil then   { Сохранение БД }
          pDBEdit(Desktop^.Current)^.DBMS^.TECommit;
      cmSaveAs      : SaveDBase;              { Сохранение как... }
      cmTile        : Desktop^.Tile(R);       { Перестройка черепицей }
      cmCascade     : Desktop^.Cascade(R); { Перестройка каскадом }
      end
  end;
end;

Procedure IDE.InitMenuBar;
Var R:TRect;
  Cur: Char;
begin
  R.Assign (0,0,80,1);       { Полоса меню вверх }
  MenuBar: = New(pMenuBar,Init(R,
  NewMenu(
      NewSubMenu('~D~atabase',hcNoContext,  { Меню БД }
      NewMenu(
          NewItem('~O~pen < 193 > ','F3',kbF3,cmOpen,hcNoContext,
          NewItem('~S~ave','F2',kbF2,cmSave,hcNoContext,
          NewItem('~S~ave as < 193 > ','',kbCtrlF2,cmSaveAs,hcNoContext,
          NewLine(
          NewItem('~E~xit','Alt-X',kbAltX,cmQuit,hcNoContext,
      nil)))))
      ),
      NewSubMenu('~W~indow',hcNoContext,   { Меню окон }
      NewMenu(
          NewItem('~S~ize/Move','Ctrl-F5',kbCtrlF5,cmResize,hcNoContext,
          NewItem('~Z~oom','F5',kbF5,cmZoom,hcNoContext,
          NewItem('~T~ile','',0,cmTile,hcNoContext,
          NewItem('~C~ascade','',0,cmCascade,hcNoContext,
          NewItem('~N~ext','F6',kbF6,cmNext,hcNoContext,
          NewItem('~P~revious','Shift-F6',kbShiftF6,cmPrev,hcNoContext,
          NewItem('~C~lose','Alt-F3',kbAltF3,cmClose,hcNoContext,
          nil)))))))
      ,nil)
  ))))
end;

procedure IDE.InitStatusLine;
var R: TRect;
begin
  GetExtent(R);
  R.A.Y : = R.B.Y - 1;
  StatusLine : = New(pNewStatusLine, Init(R,
  NewStatusDef(hcNoContext,hcNoContext,
      NewStatusKey('~F3~ Load', kbF3, cmOpen,
      NewStatusKey('~F10~ Menu', kbF10, cmMenu,
      NewStatusKey('~Alt-X~ Exit', kbAltX, cmQuit,
      NewStatusKey('',kbF5,cmZoom,
      NewStatusKey('',kbAltF3,cmClose,
      NewStatusKey('',kbCtrlF5,cmResize,
      nil)))))),
      NewStatusDef(hcDBEdit,hcDBEdit, { Контекст подсказки редактора }
      NewStatusKey('~F2~ Save', kbF2, cmSave,
      NewStatusKey('~F3~ Load', kbF3, cmOpen,
      NewStatusKey('~F4~ Edit', kbF4, cmDBEInput, { Редактирование поля }
      NewStatusKey('', kbF10, cmMenu,
      NewStatusKey('', kbAltX, cmQuit,
      NewStatusKey('', kbCtrlF7, cmDBECopyRec, { Копирование записи }
      NewStatusKey('', kbEsc, cmCancel,
      NewStatusKey('', kbSpace, cmDBEAutoInput, { Редактирование таблицы }
      NewStatusKey('', kbTab, cmDBESwitch,      { Переключение указателя }
      NewStatusKey('', kbCtrlY, cmDBEDelRec,          { Удаление записи }
      NewStatusKey('~F9~ Add', kbF9, cmDBEAddRec, { Добавление записи }
      nil))))))))))),
    NewStatusDef(hcDBEInput,hcDBEAutoInput,
                                  { Контекст при редактировании поля/таблицы }
      NewStatusKey('~Esc~ End of edit', kbEsc, cmCancel,
      NewStatusKey('~Tab/Shift-Tab~ Move', kbTab, cmDBEMoveRight,
      NewStatusKey('', kbShiftTab, cmDBEMoveLeft, { Перемещение
вправо/влево }
      NewStatusKey('', kbF10, cmMenu,
      NewStatusKey('', kbAltX, cmQuit,
      nil))))),
    nil)))));
end;

(* Not methods *)

{ Функция формирует список описателей полей из заголовка БД DBItem }
function CreateFromDB (DBItem: pDefDB):pDBHeader;
Var
  i,Len,Dec: Byte;
  ptr : ^pDBHeader;
  ptr0: pDBHeader;
  Ftype: Char;
begin
  ptr: = @ptr0;
  for i: = 1 to DBItem^.TETotalFld do
  begin
  DBItem^.TEGetFldForm (i,Ftype,Len,Dec);
  ptr^ : = NewDBField(DBItem^.TEFldName(i),FType,Len,Dec,nil);
  ptr : = @ptr^^.next
  end;
  CreateFromDB : = ptr0
end;

begin
  Appl: = New(pIDE,Init);
  Appl^.Run;
  Dispose(Appl,Done)
end.
```

Анонс

## «Монитор» — версия 5.93

**С. Курковский. Гипертексты с практической точки зрения. Часть 2**

В статье рассказывается об основах организации гипертекстовых систем. Основное внимание направлено на проблемы построения и применения гипертекстов. Предполагается опыт работы на C++. Вторая часть посвящена вопросу создания гипертекстового интерпретатора.

**Д. Арапов. Работа над ошибками**

Систематизируется опыт обработки нештатных ситуаций на Turbo Pascal.

**В. Захаров, И. Кузнецов. Turbo Vision: обнаружение некоторых ошибок в программах**

В статье рассматриваются практические приемы поиска некоторых «скрытых» ошибок в программах, создаваемых в среде Turbo Pascal 6.0 и Turbo Vision. Приведено описание некорректной работы модуля HelpFile из библиотеки Turbo Vision и даны рекомендации по устранению обнаруженных ошибок.

**Д. Валдин. Резидентные программы на языке С. Часть 1**

Статья посвящена проблемам написания резидентных программ на языке С. Описываются модели данных и особенности проектирования для компиляторов фирм Borland и Microsoft.

**А. Тихонов. Показать картинку. Часть 2**

В статье рассмотрен графический формат GIF фирмы Compuserve, приводится программа визуализации графических образов в формате GIF на Turbo Pascal.

**С. Позднеев. Применение языка Ада для разработки ПО ПЭВМ**

Рассматривается и обосновывается применение алгоритмического языка Ада и соответствующей технологии программирования для разработки программных систем на ПЭВМ. Приводятся сравнительные характеристики наиболее универсальных и распространенных систем программирования на языке Ада.

## Информация о новых вирусах в формате антивирусной программы AVSP 2.0

AVSP предоставляет пользователю возможность непосредственного ввода информации о новых вирусах в программу. Благодаря регулярной публикации такой информации AVSP на Вашем компьютере не отстанет от времени и создателей новых вирусов.

AVSP содержит ряд средств борьбы с неизвестными вирусами. ДИАЛОГ-МГУ обращается ко всем читателям с просьбой присылать информацию об обнаруженных новых вирусах.

Для ввода приведенной ниже информации в Вашу программу войдите в пункт меню «Данные о файлах и вирусах», подменю «Изменение информации о вирусах». Для более подробного описания можно обратиться к документации (пункт «Ввод информации о вирусах из журналов»).

| | |
|---|---|
| Имя вируса: | V-516 |
| Номер: | 240.1-40542 |
| Тип вируса: | PROGRAM |
| Размер: | 516 |
| Шаблон: | 5D8BF5BF0301037DFEA5 |
| Процедуры: | *.COM |
| Справка: | Заражает только те файлы, которые начинаются с команды перехода (JMP). |

| | |
|---|---|
| Имя вируса: | LoveChild |
| Номер: | 241.1-7954 |
| Тип вируса: | PROGRAM |
| Размер: | 488 |
| Шаблон: | 5E8BEEBFE001FC26813D |
| Процедуры: | *.COM { Opt(End:-459) Mov(End:-349,Beg:0,4) Tr(End:-488) } |
| Справка: | Работает только в MS-DOS 3.3. Содержит текст «LoveChild in reward for software sealing». Иногда портит EXE файлы, записывая в их начало команды порчи диска. |

По вопросам приобретения AVSP 2.0 обращайтесь в ДИАЛОГ-МГУ по телефонам:
(095) 939-38-90, 939-38-91.
Подробная информация высылается бесплатно.

Цена одного комплекта AVSP 2.0 на 1 марта 1993 г. составляет 7200 рублей (с учетом 20% налога). При покупке нескольких комплектов предоставляется скидка. Скидка для дилеров — до 60%.

# Российское научно–техническое общество РЭС им. А.С.Попова Министерства связи России приглашает фирмы и организации принять участие в крупнейшей выставке года — Международной выставке–ярмарке программных и аппаратных средств, которая состоится в Москве с 5 по 9 октября 1993 года на территории ВВЦ (бывшая ВДНХ). Выставку проводит МП «Экспосервис».

Справки по тел. (095) 921-06-59 Факс

МАГАЗИН

**+LINK**

**ЭТО ВСЁ ДЛЯ КОМПЬЮТЕРНЫХ СЕТЕЙ И КОМПЬЮТЕРОВ**

*Тел. 263-94-40* *Факс 267-58-80*

*Ново-Басманная ул., 20*

*Вы можете приобрести в нашем магазине любое оборудование к Вашему компьютеру:*

- Сетевые адаптеры ArcNet, EtherNet.
- Модемы и факс-модемы от 2400 до 14400.
- Картриджи для матричных принтеров.
- Системы бесперебойного питания UPS APC.
- Комплектующие к ПЭВМ.
- Компьютеры.
- Стриммеры.
- Сканеры и прочее.

Мы ждем Вас!

Восстанов-
ление
красящ.

Телеф.
236-88

**1. Самара**, МП «Сети», директор Дмитрий Грецов
Тел. (8462) 354-932
**2. Волгоград**, арендное предприятие «Техническая книга»
Тел. (8442) 363-597
**3. Чебоксары**, фирма «Демос»
Тел. (8352) 208-395, 426-492, факс 209-437
**4. Киев**, МП «Диас»
Тел./факс (044) 220-77-15, тел. 518-80-24
**5. Днепропетровск**, Попов А.Е.
Тел. (0562) 428-593
**6. Минск**, НПП «Триумф»
Тел. (0172) 661-609, 661-680, факс 666-606
**7. Ташкент**, Центр молодежных исследований
Тел. (3712) 688-847
**8. Тамбов**, АО «От Елены» ЛПС «Shift»
Тел. (0752) 22-44-63
**9. Пермь**, Попов Андрей Леонидович
Тел. (3422) 33-30-91
**10. Иркутск**, Мадьяров Игорь Иосифович
Тел. (3952) 46-19-68

# Уважаемые читатели!

Мы предлагаем Вашему вниманию

# компьютерный "ларек".

Мы преследуем две основные цели:

– дать возможность отечественным разработчикам инструментальных средств выйти на российский рынок программного обеспечения;

– обеспечить как можно более широкий круг читателей недорогими и оригинальными средствами разработки программного обеспечения.

| Код | Продукт | Краткая характеристика | Объем поставки | Цена, $ | Требования к оборудованию | Поддержка | Число продаж |
|---|---|---|---|---|---|---|---|
| SL-01 | Turbo Games | Объектно-ориентированная библиотека для Turbo Pascal 6.0. Спрайты, звуковые эффекты, трехмерная графика, стереоэффекты. Доступ к ZIP файлам. | 2 дискеты + документация | 12 для индивидуальных покупателей, 25 для организаций | EGA адаптер, IBM PC/AT | + | 27 |
| SL-02 | Data Vision 1.5 | Объектно-ориентированная библиотека для Turbo Pascal 6.0. Создание и управление реляционными базами данных. Поддерживаются DBF и DB форматы, а также электронные таблицы. | 1 дискета + документация | 25 | IBM PC/XT | + | 47 |
| SL-03 | AtoII | Библиотека для Turbo C. Динамический просмотр и масштабирование графических изображений. | 2 дискеты + документация | 10 | IBM PC XT/AT, EGA/VGA, PCX-system | + | новый |
| SL-04 | SPRight Lib | Объектно-ориентированная библиотека для Borland C++ 1.xx-2.xx. Спрайты, спрайтовые объекты, звуковые эффекты, поддержка графического формата PCX. Прозрачна для большинства графических библиотек. | 1 дискета + документация | 15 | EGA/VGA адаптер, IBM PC/AT | + | новый |
| IS-01 | HyperRef | Инструментальное средство для создания гипертекстовых систем. Поддержка текстовой информации, изображений и исполняемых программ. | 2 дискеты | 70 | IBM PC/AT, EGA/VGA, 3 Мб на HDD, 350 Кб ОЗУ — просмотр документа, 450 Кб — редактирование документа. | | 1 |
| SC-01 | QuickTools 1.0 | Средство разработки диалога в Paradox. | 1 дискета + документация | 12 | IBM PC/AT, EGA/VGA, 3 Мб на HDD, 1-2 Мб ОЗУ. | + | 8 |
| SC-02 | QB Tools 3.0 | Средство разработки диалога, а также поддержки DBF формата для Quick и Professional Basic. | 1 дискета | 30 для индивидуальных покупателей, 50 для организаций | IBM PC/AT, EGA, 3 Мб на HDD, 640 Кб ОЗУ | + | новый |
| UT-01 | ArcView 6.1 | Просмотр и редактирование заголовков архивных файлов с возможностью подключения собственных типов архивов. | 1 дискета 360 Кб | 1 | IBM PC/AT, Hercules, EGA/VGA | + | |
| SE-01 | Crypter 2.0 | Система шифрования файлов. Поддерживаются алгоритмы DES, ГОСТ. | 1 дискета 360 Кб | 12 | IBM PC/AT, Hercules, EGA/VGA | + | |
| SC-01 | АвтоГРАФ | Пакет для создания на экране графиков с результатами обработки числовых данных и печати их на 9 и 24-игольчатых принтерах. Поддержка форматов ЕСКД. | 1 дискета 360 Кб | 10 | IBM PC/XT, EGA/VGA, 500 Кб на HDD, ОЗУ — 200 Кб | + | 73 |

*Если Вас заинтересовали программные продукты или Вы хотите получить дополнительную информацию, звоните, пишите, приезжайте.*

*Оплата принимается только в рублях по текущему курсу доллара.*

*Адрес для писем: 117419 Москва, а/я 765.*

*Тел.: (095) 179-50-69 (Морев Александр). Тел./факс: (095) 238-99-33, 238-31-89, 238-32-78.*

# Инструментарий QuickTools для создания прикладных систем в среде PARADOX

*С. Саркисян,*
*Д. Шляйфер,*
*Л. Ливеровский*

Прежде чем перейти к обсуждению подходов, используемых при создании инструментальных средств разработки прикладных систем, попытаемся конкретизировать предметную область, а также проанализировать возникающие при этом проблемы и возможные пути их решения.

Итак, что мы будем понимать под термином «прикладная система»? Обычно под прикладной системой понимается программное обеспечение, предназначенное для решения определенной задачи или класса задач или для предоставления пользователю определенных услуг. Под это широкое определение попадают различные прикладные системы, обслуживающие самые разные классы пользователей, от совершенно неподготовленных до специально обученных. Среди них не только системы, разрабатываемые годами и распространяемые десятками или даже сотнями тысяч экземпляров, но и те, что существуют в единственном экземпляре. Объектом нашего внимания будут именно такие, часто называемые заказными системы, разрабатываемые по специальному заказу и существующие в одном или нескольких экземплярах.

Среди множества заказных систем мы будем рассматривать те, которые используют некоторую СУБД и ориентированы на определенный класс пользователей — специалистов в своей предметной области, не являющихся профессионалами в области обработки данных и применения средств вычислительной техники. Это накладывает целый ряд требований к проектированию систем.

1) Разработка заказной системы требует определения заказчиком возможно более точных спецификаций, однако практика показывает, что получить их практически не удается.

2) Малое количество копий системы не позволяет покрыть издержки на ее разработку за счет тиражирования, поэтому затраты на разработку должны быть минимальны, а сроки разработки ограничены.

3) Внешний интерфейс системы должен быть простым, интуитивно понятным и дружественным по отношению к пользователю. Он чаще всего представляет собой совокупность меню, обеспечивающих пользователю выбор одной из ряда альтернатив. Поскольку общение с системой происходит на русском языке, в меню нужно использовать достаточно полные наименования альтернатив (сокращения, как правило, ухудшают восприятие).

4) Документация на прикладную систему, по крайней мере, в части, являющейся инструкцией

по работе с системой, должна максимально соответствовать тому, что пользователь может увидеть на экране в процессе эксплуатации.

Перечисленные особенности определяют ряд требований уже к инструментальным средствам, предназначенным для проектирования заказных систем.

Во-первых, в постановке задачи совершенно необходимо участие заказчика, в роли которого выступает конечный пользователь, не являющийся специалистом в области обработки данных и проектирования систем на базе средств вычислительной техники. Наилучшим решением представляется наличие инструментальных средств, позволяющих заказчику самому строить действующий макет системы, содержащий большую часть меню, вызывающих функции обработки данных. Это также обеспечивает важный психологический фактор участия заказчика в создании системы и резко сокращает сроки создания технического проекта системы.

Во-вторых, инструментальные средства должны гарантировать кратчайшие сроки создания системы, а также минимизацию затрат на кодирование, отладку, тестирование и проведение доработок.

В-третьих, интерфейс самих инструментальных средств должен удовлетворять тем же требованиям, что и интерфейс проектируемой заказной системы.

Рассмотрим далее различные подходы к решению проблемы создания инструментальных средств для разработки заказных систем. Традиционно существуют два основных подхода: процедурный и непроцедурный. Первый подход основан на использовании универсального (такого, как C или C++) либо проблемно-ориентированного (например, COBOL или PAL) языка программирования, включающего библиотеки специализированных процедур, упрощающих работу в выбранной предметной области. Этот подход, безусловно, обеспечивает большую гибкость в разработке и более высокое быстродействие (что особенно важно в интерфейсной части) создаваемых прикладных систем. Недостатками процедурного подхода являются большая нагрузка на программистов-профессионалов, сложность привлечения заказчика к созданию действующего макета системы, а также длительные сроки разработки.

Непроцедурный подход в основе рассчитан на пользователя-непрограммиста, сообщающего инструментальной системе не КАК, а ЧТО необходимо сделать. Одной из разновидностей такого подхода является известный принцип WYSIWYG (What You See Is What You Get), иначе называемый визуальным программированием. Он позволяет создавать объекты прикладной системы и устанавливать между ними связи, после чего инструментальная система автоматически создает их описания, интерпретируемые в процессе выполнения, либо генерирует программный код на некотором базовом языке. Непроцедурный подход в большей степени соответствует перечисленным выше требованиям к инструментальным системам, однако и он не свободен от недостатков, так как всегда ограничен возможностями конкретной инструментальной системы, выход за рамки которой часто невозможен или затруднителен. Причина этого очевидна: гораздо проще построить дом из крупных панелей, чем из кирпичей, но попробуйте перенести в нем дверь или окно на другое место — придется пробивать стену.

Идеальной является инструментальная система, соединяющая оба подхода: для проектирования объектов в ней должен использоваться принцип WYSIWYG, в то же время должны быть предусмотрены непосредственное включение в генерируемый код операторов базового языка и обширная библиотека специализированных процедур на базовом яыке.

Как было сказано, весьма обширной областью разработки заказных систем являются задачи обработки данных. Для их решения чаще всего выбирается одна из существующих популярных СУБД. В частности, авторы этой статьи в течение долгого времени занимались разработкой заказных систем на базе СУБД Paradox сначала версии 3.0, а затем и 3.5 (в настоящее время уже появилась версия 4.0).

Оценивая инструментальные средства Paradox, следует отметить наличие в их составе средств проектирования форм ввода данных, отчетов и запросов (в качестве языка запросов используется вариант QBE — Query By Example). Имеется также собственный проблемно-ориентированный язык PAL (Paradox Application Language), содержащий средства вывода на экран типовых меню Paradox (горизонтальных, расположенных в верхней строке экрана), а также средства организации диалога. Paradox 4.0 имеет более широкие возможности в части организации диалогового интерфейса, однако требует для работы не менее 2 Мб оперативной памяти (для эффективного функционирования требуется около 4 Мб).

Для автоматизации разработки интерфейсной части прикладных систем в Paradox предусмотрена специальная утилита — Personal Programmer. Однако с ее помощью можно создавать лишь те же архаичные горизонтальные меню, причем вложенные меню замещают на экране меню, из которых они были вызваны. Как известно, русский язык не столь краток, как английский, поэтому, чтобы разместить альтернативы на экране, приходится ограничивать их количество в одном меню, увеличивая число меню и нарушая логику распределения альтернатив. Другими недостатками Personal Programmer являются невозможность управления цветом меню и связывания с одной альтернативой более одного действия.

Слабость Personal Programmer, видимо, сознаваемая разработчиками, заставила их включить в поставляемую вместе с Paradox библиотеку процедур Toolkit процедуры PopUp и PopUp2, позволяющие динамически формировать на экране вертикальные меню. Информация об альтернативах таких меню хранится в обычных таблицах СУБД Paradox и переносится в память специальными программами SetPopUp и SetPopUp2, включенными в ту же биб-

лиотеку. К сожалению, эти программы работают довольно медленно, и, кроме того, задача связывания альтернативы с соответствующим действием ложится на плечи программиста. Справедливости ради следует отметить, что соответствующий инструмент в Paradox 4.0 (утилита Workshop) имеет несколько большие возможности в плане организации интерфейса, но и здесь они достаточно ограничены.

Язык программирования PAL также имеет ряд недостатков. Во-первых, есть лишь интерпретатор языка PAL. Скорость интерпретации составляет около 1000 оп/сек, а операторы вывода на экран интерпретируются в 5 — 7 раз медленнее (данные получены на компьютере с процессором 386SX/16 МГц и памятью 2 Мб).

Во-вторых, PAL не имеет средств считывания с экрана, следовательно, экран невозможно сохранить (операторы PAL взаимодействуют с двумя экранами, называемыми WorkSpace и Canvas). Средствами языка можно лишь выводить текст и раскрашивать Canvas, но нельзя ничего считать с него.

В-третьих, хотя в языке и имеются массивы, идентификатор массива нельзя передать в качестве формального параметра.

В-четвертых, в PAL отсутствует строгая типизация данных, и переменная даже в рамках одной программы может принимать значения разных типов.

Все это вместе взятое создает ряд сложностей при программировании на языке PAL, столкнувшись с которыми при разработке заказных систем, авторы статьи создали собственную инструментальную систему, которая носит название QuickTools for Paradox. В настоящее время распространяется версия 1.0 этой системы, обеспечивающая создание интерфейсной части прикладных систем.

QuickTools позволяет проектировать меню различных типов: горизонтальные, вертикальные и матричные, предлагая при этом широкий выбор средств оформления меню (заголовки, окаймления, раскраски, подсказки и так далее). Для каждой альтернативы меню можно указать имя инструкции (Help), которая будет выведена при нажатии клавиши <F1>. Одновременно на экране может присутствовать любое количество меню, которые можно размещать на экране индивидуально, явно задавая координаты меню либо указывая правила, по которым они определяются, в зависимости от точки вызова меню. Наконец, QuickTools позволяет управлять такими элементами оформления интерфейса, как фон, на котором появляются меню, а также вывод титульного окна экрана и заставок. Мощность и гибкость средств QuickTools подтверждается тем обстоятельством, что они использовались для создания собственного интерфейса QuickTools.

Функционально QuickTools включает ряд подсистем, каждая из которых представляет набор программ, написанных на языке PAL, а именно:
- подсистему разработки меню;
- подсистему разработки инструкций (Help);
- подсистему разработки заставок;
- генератор прикладных систем.

Помимо подсистем, в QuickTools входят библиотеки процедур, которые реализуют типовые функции обработки данных и работы с экраном и могут быть использованы при программировании на языке PAL, сокращая сроки разработки приложений.

Все объекты проектирования (то есть меню, заставки, инструкции) и их совокупности (или прикладные системы) размещаются в некоторой области на жестком диске — рабочем каталоге. QuickTools ведет список всех рабочих каталогов и всех объектов, находящихся в них. Любой из объектов может копироваться из одного каталога в другой, при этом можно свободно переписывать меню из одной прикладной системы в другую.

Основным этапом проектирования является создание меню, осуществляемое в QuickTools в режиме WYSIWYG. Каждое меню однозначно определяется типом меню и описывается с помощью альтернатив и атрибутов меню. Управление активным в настоящий момент меню осуществляется специальным компонентом QuickTools, называемым драйвером меню. Для каждого типа меню предусмотрен свой драйвер, обеспечивающий реакцию на действия пользователя и перемещающий окошко выбора по альтернативам меню.

С каждой из альтернатив меню можно связать любое количество элементарных действий, выполнение какого-либо сценария, вызов подчиненного меню, выход из меню, вывод на экран инструкции (Help), фиксация выбора альтернативы, выполнение команд PAL. Последнее свойство позволяет непосредственно включать в код реализации меню операторы и функции языка PAL, обеспечивая дополнительную гибкость при создании приложений.

Пример привязки действий к элементу меню в среде дизайнера меню QuickTools показан на рис. 1.

Атрибуты оформления меню не являются обязательными, к ним относятся: заголовок меню, окантовка меню, цвета меню, разделительные строки и надписи, размещение на экране альтернатив и самой области меню, действия с экраном за границами этой области, правила выхода из меню (устанавливающие, например, автоматический выход после выбора какой-либо альтернативы и выполнения связанных с ней действий, блокировку или разрешение выхода по <Esc>, необходимость запроса подтверждения при попытке выйти из меню и так далее).

Все параметры меню хранятся в таблице, фрагмент которой для разбираемого примера приведен в табл. 1.

Заставкой называется специально оформленный текст, который может быть использован для вывода специальных сообщений, предваряющих какие-либо действия прикладной системы, либо маскирующих выполнение длительной обработки данных. Средства разработки заставок QuickTools позволяют быстро создавать заставки типа той, которую можно увидеть на экране в момент запуска СУБД Paradox.

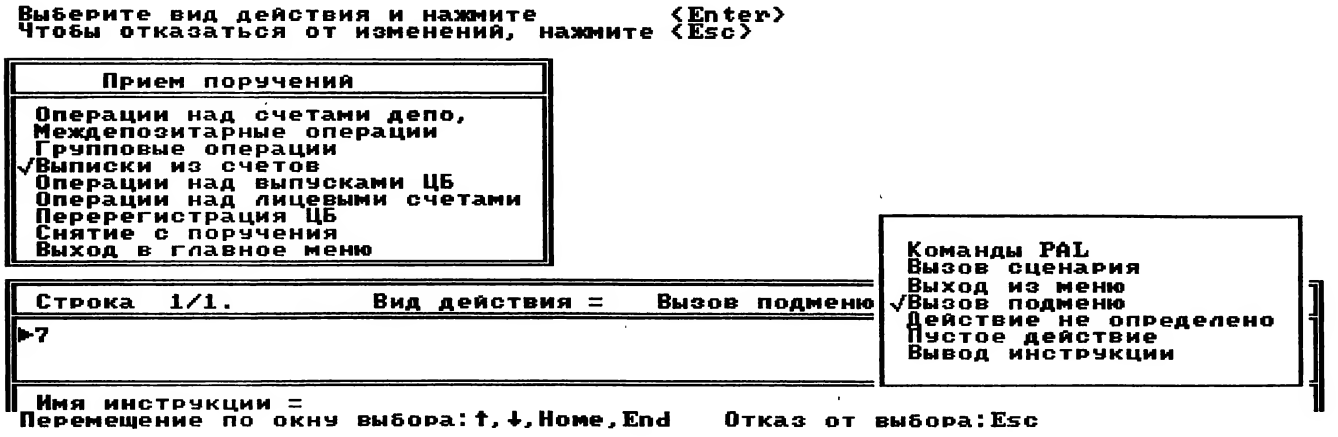



**Рис. 1. Привязка действий к элементу меню в среде дизайнера меню QuickTools**

Инструкции являются одним из наиболее важных элементов интерфейса прикладной системы. Хорошо продуманная контекстно-чувствительная система инструкций должна обеспечивать пользователя всей необходимой ему информацией именно тогда, когда он в ней нуждается. Инструкции в QuickTools являются самостоятельными объектами проектирования, которым присваиваются идентифицирующие их имена. QuickTools предлагает несколько способов организации системы инструкций.

Во-первых, с каждой альтернативой любого меню можно связать инструкцию, которая вызывается нажатием клавиши <F1> в момент, когда этот элемент появляется в окошке меню.

Во-вторых, для определенных альтернатив меню можно в качестве одного из действий определить вызов инструкции на экран.

В-третьих, при программировании можно использовать специальную процедуру приема очередного символа, обеспечивающую вызов инструкции при нажатии определенных клавиш или их комбинаций.

Помимо собственных средств QuickTools, для создания инструкций можно использовать любой внешний текстовый редактор, импортируя затем тексты Help с помощью специальной компоненты QuickTools.

QuickTools позволяет определить ряд дополнительных атрибутов прикладной системы, таких как оформление экрана, наличие начальной заставки системы, сведения о компьютере, на котором будет выполняться прикладная система (целевой машине). Гибкие средства настройки, включенные в QuickTools, позволяют создавать приложения, которые могут выполняться на машинах с оперативной памятью до 1 Мб, с черно-белым или цветным монитором, работать в сети или на отдельной машине.

Спроектированная прикладная система с помощью подсистемы QuickTools «генератор прикладных систем» превращается в процедуры (сценарии) на языке

**Таблица 1**

| N menu | Type | Text | Help | Act | Parm |
|---|---|---|---|---|---|
| | S | N=DEPOZ;F=;FC=23;W=0,0,80;TC=31; | | | |
| | S | T=Система "ДЕПОЗИТАРИЙ" | | | |
| | S | L=d:\\desnrkit\\asgdlib | | | |
| | S | U=appname,menuappname,extract | | | |
| ..... | | | | | |
| ..... | | | | | |
| 2 | A | F=CVAA17WNNNN,S=31,H=0,BC=113,R=2,C=2 | | | |
| 2 | A | W= | | | |
| 2 | Z | Прием поручений | | | |
| 2 | T | Операции над счетами депо | | M | 5 |
| 2 | T | Междепозитарные операции | | M | 6 |
| 2 | T | Групповые операции | | M | 8 |
| 2 | T | Выписки из счетов | | M | 7 |
| 2 | T | Операции над выпусками ЦБ | | M | 9 |
| 2 | T | Операции над лицевыми счетами | | M | 10 |
| 2 | T | Перерегистрация ЦБ | | M | 11 |
| 2 | T | Снятие с поручения | | N | |
| 2 | T | Выход в главное меню | | O | |

PAL, которые могут быть оттранслированы и записаны в библиотеку (поскольку в PAL имеется лишь интерпретатор, трансляция здесь означает предварительный синтаксический разбор и формирование так называемого массива интерпретации). Готовая прикладная система может выполняться как под управлением СУБД Paradox, так и под управлением системы Paradox Runtime.

Внешний вид главного экрана QuickTools представлен на рис. 2.

К сожалению, размеры статьи не позволяют более подробно рассказать обо всех возможностях QuickTools, поэтому мы ограничимся уже сказанным.

Спроектированные объекты с помощью генератора прикладных систем превращаются в код на языке PAL. Мы не будем подробно излагать методику перевода описания объектов в код, остановимся лишь на некоторых технических решениях, обеспечивающих необходимое быстродействие и продиктованных невысоким быстродействием интерпретатора, а также отсутствием средств сохранения и восстановления экрана.

Каждому меню соответствуют две генерируемых процедуры, одна из которых обеспечивает визуализацию меню на экране, а вторая — отслеживание перемещений по меню и активизацию действий в соответствии с выбором альтернатив. Программа визуализации меню содержит в себе заранее подготовленное изображение меню в виде блока Text...EndText, являющееся наиболее быстрым способом вывода на экран. Отсутствие в PAL средств сохранения и восстановления экрана преодолевается наличием специального механизма, суть которого состоит в следующем: главная программа прикладной системы определяет массив Trace (стек) размером в 105 элементов и указатель этого массива с именем Ltrace. Каждый элемент стека занимает 7 элементов массива Trace, первый из которых содержит имя процедуры визуализации некоторого объекта, а остальные 6 — дополнительную информацию, которая может использоваться при визуализации. При входе в программу управления каким-либо меню в стек заносится соответствующая информация, которая удаляется при выходе из этой программы. Заметим, что в стек можно поместить информацию о визуализации любого объекта, а не только меню.

При необходимости восстановить экран запускается библиотечная процедура TraceDown, которая автоматически включается в состав любой прикладной системы и обеспечивает восстановление экрана, запуская присутствующие в стеке программы визуализации объектов. Восстановление экрана с заполнением, титульным окном и 5 меню по измерениям, проведенным авторами, заняло около 0,25–0,3 сек. В листинге 1 приведен пример сгенерированной программы визуализации меню, а в листинге 2 — программа управления меню.

Подведем некоторые итоги. Подход, описанный в статье, позволил авторам создать простую в эксплуатации и вместе с тем мощную инструментальную систему, обеспечивающую проектирование в режиме WYSIWYG и осваиваемую непрофессионалами, как показала практика, за считанные часы. Гибкость приложений обеспечивается, с одной стороны, наличием развитой библиотеки модулей, а с другой — возможностью включения в генерируемый код операторов языка PAL.

Направлениями последующего развития возможностей системы QuickTools являются включение в систему средств проектирования динамически формируемых меню, позволяющих перемещением подсветки выбирать нужные объекты из выведенной на экран совокупности, а нажатием функциональных клавиш задавать операции над этими объектами, средств проектирования систем ввода данных, обеспечивающих ввод исходных данных в сгенерированные формы ввода, их контроль и последующую разноску по заданным полям таблиц базы данных, и, наконец, создание словаря баз данных, облегчающего проектирование сложных прикладных систем и обеспечивающего принципиальную возможность генерации не только интерфейсных компонентов приложений, но и компонентов, осуществляющих доступ и обработку содержимого баз данных. ■

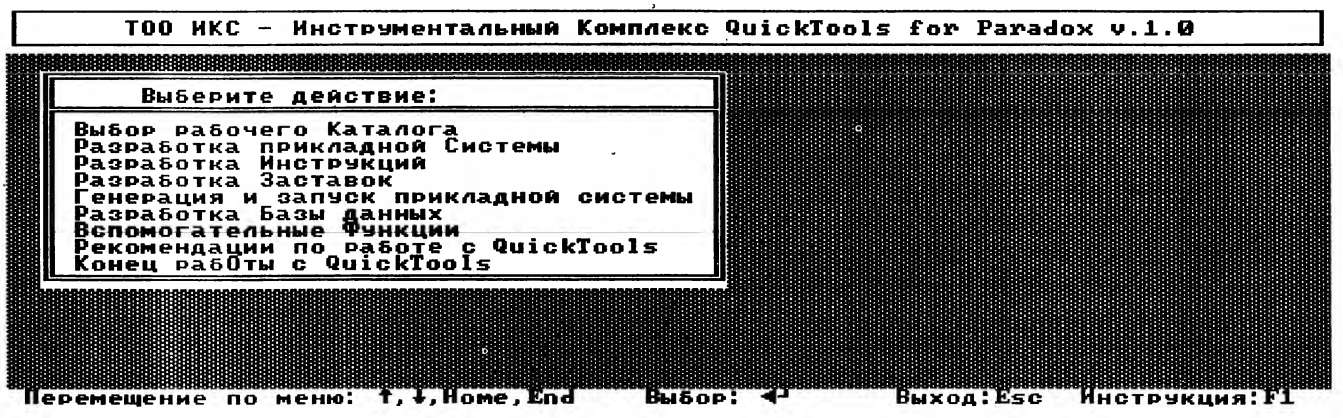



**Рис. 2. Внешний вид главного экрана QuickTools**

## Листинг 1. Сгенерированная программа визуализации меню

```
Proc DEPOZM2()
;*****************************************************
;Эта процедура обеспечивает визуализацию меню некоторой
; прикладной системы DEPOZ, меню имеет номер 2.
;*****************************************************
Canvas off                    ;Блокировка непосредственного вывода
                              ;на экран. Операторы вывода заносят
                              ;данные в образ экрана в памяти.
If TraceOff Then              ;Переключатель TraceOff показывает,
                              ;установлен режим восстановления
                              ;экрана или режим вывода меню.
    If not MoveFrd Then       ;Если установлен режим вывода меню и
        Ltrace = Ltrace-6     ;меню на экране восстанавливается
        TraceDown()           ;после изменения экрана, запускается
        Lrace = Ltrace+6      ;процедура восстановления
    Endif                     ;предшествующих объектов на экране.
Endif
Style Attribute 31            ;Установка цветов вывода текста.
SetMargin Trace[Ltrace+3]     ;Установка координат точки привязки
@ Trace[Ltrace+2],Trace[Ltrace+3]   ;области меню, они берутся из стека.
Text
```

```
  Прием поручений

  Операции над счетами депо
  Междепозитарные операции
  Групповые операции
  Выписки из счетов
  Операции над выпусками ЦБ
  Операции над лицевыми счетами
  Перерегистрация ЦБ
  Снятие с поручения
  Выход в главное меню
```

```
Endtext                       ;Текст меню выведен.
Style Attribute 30            ;Установка цветов букв, по которым
                              ;можно выбрать альтернативы меню.
@ Trace[Ltrace+2]+3,Trace[Ltrace+3]+2 ?? "О"
@ Trace[Ltrace+2]+4,Trace[Ltrace+3]+2 ?? "М"
@ Trace[Ltrace+2]+5,Trace[Ltrace+3]+2 ?? "Г"
@ Trace[Ltrace+2]+6,Trace[Ltrace+3]+2 ?? "В"
@ Trace[Ltrace+2]+7,Trace[Ltrace+3]+25 ?? "Ц"
@ Trace[Ltrace+2]+8,Trace[Ltrace+3]+15 ?? "Л"
@ Trace[Ltrace+2]+9,Trace[Ltrace+3]+2 ?? "П"
@ Trace[Ltrace+2]+10,Trace[Ltrace+3]+2 ?? "С"
@ Trace[Ltrace+2]+11,Trace[Ltrace+3]+2 ?? "К"
                              ;Подкраска букв закончена.
If Trace[Ltrace+1] < > "" Then    ;Если восстанавливается меню,
                              ;которое уже было на экране,
                              ;в этом элементе стека находятся
                              ;координаты и длина "окошка" меню,
                              ;наложенного на какую-либо
                              ;альтернативу.
    PaintCanvas Attribute 48  ;Подкрасить альтернативу в окошке.
    int(Trace[Ltrace+1]/10000),
    mod(int(Trace[Ltrace+1]/100),100),
    int(Trace[Ltrace+1]/10000),
    mod(int(Trace[Ltrace+1]/100),100)+mod(Trace[Ltrace+1],100)
Endif
SetMargin Off
                              ;Все сделано. Собственно вывод на
                              ;экран будет осуществлен, когда
Endproc                       ;управление получит драйвер меню.
```

## Листинг 2. Программа управления меню

```
Proc DEPOZMenu2(DEPOZR,DEPOZC)
;*****************************************************
;Эта программа осуществляет управление действиями,
;связанными с меню.
;В примере действия очень просты и сводятся к вызову подчиненных
;меню или к выходу из данного меню. Меню вертикального типа с
;автоматической привязкой к точке его вызова. Параметрами
;программы являются значения координат точки вызова меню.
;*****************************************************
;Глобальные переменные:
;Trace          -   стек следа активных меню на экране;
;Ltrace         -   указатель стека;
;TraceOff       -   значение False показывает, что установлено
;                   состояние восстановления экрана;
;MoveFrd        -   значение False показывает, что был выход из
;                   подчиненного меню либо содержимое
;                   экрана изменилось;
;CnvasChngd     -   значение True показывает, что содержимое экрана
;                   было изменено.
;*****************************************************
;Локальные переменные:
Private DEPOZD,               ;номер альтернативы меню в окошке;
    zzzD,                     ;рабочая переменная;
    MenuKoord,                ;массив относительных координат и
                              ;длин альтернатив меню;
    MenuHlp,                  ;массив имен инструкций,
                              ;соответствующих альтернативам меню;
    MenuBlockEsc,             ;признак блокировки выхода из меню
                              ;при нажатии <ESC>;
    MenuLetCol,               ;цвет подкрашенных символов в
                              ;альтернативах меню;
    MenuLetPos,               ;позиции подкрашенных символов;
    MenuLetStr1,              ;значения подкрашенных символов на
    MenuLetStr2               ;верхнем и нижнем регистрах.

;*****************************************************
Cursor Off                    ;Отключить изображение курсора;
If Trace[Ltrace+6] < > "V" Then   ;смещение от точки вызова по
    DEPOZR = DEPOZR+1         ;столбцу, в зависимости от типа
Endif                         ;вызывающего меню,
If Trace[Ltrace+6] = "V" Then     ;смещение от точки вызова по
    DEPOZC = DEPOZC+mod(Trace[Ltrace+1],100) ;строке, в зависимости
Endif                         ;от типа вызывающего меню;
DEPOZR = max(0,min(DEPOZR,10))    ;если необходимо, исправить
DEPOZC = max(0,min(DEPOZC,39))    ;координаты верхнего левого угла
                              ;области меню так, чтобы оно
                              ;разместилось на экране;
Array MenuKoord[9]            ;определить массив координат и
Array MenuHlp[9]              ;инструкций для элементов меню,
MenuKoord[1] = 30232          ;задать соответствующие значения.
MenuKoord[2] = 40224
MenuKoord[3] = 50218
MenuKoord[4] = 60217
MenuKoord[5] = 70235
MenuKoord[6] = 80229
MenuKoord[7] = 90228
MenuKoord[8] = 100218
MenuKoord[9] = 110235
MenuHlp[1] = "Pp2Dp"
MenuHlp[2] = "Pp2Md"
MenuHlp[3] = "Pp2Go"
MenuHlp[4] = "Pp2Vo"
MenuHlp[5] = "Pp2Vb"
MenuHlp[6] = "Pp2Lo"
MenuHlp[7] = "Pp2Nr"
MenuHlp[8] = "Pp2Un"
MenuLetCol = 30
MenuLetPos = "\001\001\001\001\024\014\001\001\001"
MenuLetStr1 = "ОМГВЦЛПСК"
MenuLetStr2 = "омгвцлпск"
;Регистрация входа в меню в стеке
Ltrace = Ltrace+7
Trace[Ltrace] = "DEPOZM2"
Trace[Ltrace+1] = ""
Trace[Ltrace+2] = DEPOZR
Trace[Ltrace+3] = DEPOZC
Trace[Ltrace+4] = DEPOZR+13
Trace[Ltrace+5] = DEPOZC+39
Trace[Ltrace+6] = "V"
MoveFrd = TRUE
DEPOZD = 1
zzzD = DEPOZD
DEPOZM2()                     ;Визуализация меню;
MenuBlockEsc = False          ;для меню выход по <ESC> разрешен;
While TRUE                    ;основной цикл, выход из которого
                              ;выполняется при нажатии <ESC> или
                              ;выборе альтернативы, связанной
                              ;с действием "выход из меню";
If(CnvasChngd or (not MoveFrd))   ;если зачем-либо нужно
    Then                      ;восстановить экран, снова
        MoveFrd = False       ;выполняется программа
        DEPOZM2()             ;визуализации меню;
        CnvasChngd = False
Endif
DEPOZD = DrV(DEPOZR,DEPOZC,9,31,DEPOZD,48,"",0)
                              ;запуск драйвера;
If DEPOZD > 0 Then
    MoveFrd = True            ;пользователь выбрал альтернативу,
                              ;выяснить, какую и активизировать
    Switch                    ;действие.
        Case DEPOZD = 1:      ;Операции над счетами депозитария,
            DEPOZMenu5(DEPOZR+3,DEPOZC+2)   ;вложенное меню;
        Case DEPOZD = 2:      ;Междепозитарные операции,
            DEPOZMenu6(DEPOZR+4,DEPOZC+2)   ;вложенное меню.
        Case DEPOZD = 3:      ;Групповые операции,
            DEPOZMenu8(DEPOZR+5,DEPOZC+2)   ;вложенное меню.
        Case DEPOZD = 4:      ;Выписки из счетов,
            DEPOZMenu7(DEPOZR+6,DEPOZC+2)   ;вложенное меню.
        Case DEPOZD = 5:      ;Операции над ценными бумагами,
            DEPOZMenu9(DEPOZR+7,DEPOZC+2)   ;вложенное меню.
        Case DEPOZD = 6:      ;Операции над лицевыми счетами;
            DEPOZMenu10(DEPOZR+8,DEPOZC+2)  ;вложенное меню.
        Case DEPOZD = 7:      ;Перерегистрация ценных бумаг;
            DEPOZMenu11(DEPOZR+9,DEPOZC+2)  ;вложенное меню.
        Case DEPOZD = 8:      ;Снятие с поручения;
            Message "Действие не определено"
            Sleep 2000
        Case DEPOZD = 9:      ;Выход в главное меню;
            Quitloop
    EndSwitch
Else                          ;Была нажата <ESC>, выйти из меню.
    retval = ""
    Quitloop
Endif
Endwhile
;*****************************************************
MoveFrd = False
Trace[Ltrace+1] = ""          ;Убрать запись о меню из стека,
Ltrace = Ltrace-7             ;закончить работу программы.
Endproc
```

# О Clipper в стиле «ретро»

*В. Хорт*

При разработке программ с использованием системы Clipper-87 убеждаешься, что возможности этого языка, по меткому выражению одного из программистов, «широки и поистине безграничны». Красивые программы и сравнительно компактные исполнительные модули, полученные после компиляции текста программы, поначалу удовлетворяют многих. Спустя некоторое время, впрочем, появляется ощущение, что все хорошо, но вот именно это можно было бы сделать проще, если бы была простенькая функция, реализующая то-то и то-то. Чаще всего для придания программам законченного вида не хватает возможностей Clipper по использованию «мыши» и клавиш-модификаторов Ctrl, Alt и Shift. Решить такие проблемы можно, обходя многочисленных знакомых в поисках нужных расширений к Clipper, или отказаться от попыток достичь высот Питера Нортона, или, что по мнению автора самое полезное, пойти в магазин и купив, например, журнал «Монитор», раз и навсегда разобраться с проблемами написания собственных функций.

Когда в первом номере журнала «Компьютеры+Программы» за 1992 год была напечатана короткая программа [1], позволяющая выводить в одной из строк экрана разные подсказки, в зависимости от того, нажата клавиша Shift, Ctrl, Alt или нет, я решил использовать эту возможность в своих программах. Учитывая, что мое знание языка C на тот момент оставляло желать лучшего, пришлось приступить к поиску компилятора Quick C версии 2.5, как было рекомендовано в статье. В процессе поиска компилятора и часто выслушиваемых соображений о том, что лучше использовать Turbo C, удалось собрать обломки нужного пакета и приступить к трансляции текстов, опубликованных в журнале. Поскольку программа не желала транслироваться, пришлось приступить к более основательному изучению C. Первые же шаги в изучении языка позволили найти ошибку в тексте программы, опубликованной в упомянутом журнале (обращаю внимание авторов названной статьи, что ошибка не в программе, а в опубликованном тексте!). Исправление ошибки позволило откомпилировать полученную программу в Turbo C, «одолжив» у Quick C лишь два файла:

nandef.h и extend.h.

В процессе освоения программы обработки клавиш Ctrl, Alt и Shift стало ясно, как можно создавать свои дополнительные функции для Clipper. Поскольку в нашей стране более распространены пакеты C фирмы Borland, то, дополнив их упомянутыми модулями, можно приступать к работе.

Прежде чем переходить к созданию дополнительных функций для работы с «мышью», вспомним, как компьютер работает с этим устройством. После установки драйвера «мыши» включением/выключением курсора, отслеживанием его координат и состоянием кнопок ведает прерывание INT 33h. При этом, в зависимости от содержимого регистра AX, можно, обращаясь к этому прерыванию, добиваться различных результатов (подробнее см., например, [2]).

Так, обращаясь к прерыванию 33h после засылки в AX нуля, узнают, активизирован или нет драйвер «мыши». После такого обращения к прерыванию регистр BX содержит 0, если «мыши» нет, и количество кнопок — в противном случае. Обращаясь к INT 33h со значением AX=1, курсор «мыши» делают видимым, а со значением AX=2 — невидимым. После обращения к прерыванию со значением AX=3 по содержимому регистров CX и DX определяют горизонтальную и

вертикальную координаты курсора «мыши» соответственно, а по содержимому BX — состояние кнопок (0 — ни одна из кнопок не нажата, 1 — нажата левая кнопка, 2 — нажата правая кнопка, 4 — нажата средняя кнопка).

Теперь можно приступать к написанию дополнительных функций для Clipper. Пример одной из таких функций приведен в листинге 1. В нем вводятся дополнительные функции включения/выключения курсора «мыши» — mouse_on() и функция mouse_key(), позволяющая определить координаты курсора «мыши» и состояние кнопок.

Функция _retni(int) (описана в extend.h) осуществляет возврат в Clipper программу с передачей одного целого числа. Тем не менее, в этом числе содержится информация о трех параметрах: вертикальной координате курсора «мыши» — y, горизонтальной координате — x и о состоянии кнопок — key. Эта информация шифруется следующим образом:

```
rez = 1000 * y + 10 * x + key.
```

Поскольку возврат в Clipper нельзя осуществлять с передачей более чем одного параметра, то приходится зашифровывать нужную информацию в одно число. Порядок, в котором осуществляется шифрование — единственный, при котором возвращается целое число типа int, так как значение y лежит в пределах от 0 до 24, x — от 0 до 79, key — от 0 до 4. Можно возвращаться, используя функцию _retnl(long), тогда возможна передача целых чисел типа long.

При работе функции mouse_on() определяется количество параметров, переданных ей при обращении из программы на Clipper посредством переменной PCOUNT, а значение параметров — при помощи функции _parni() (PCOUNT и _parni() также описаны в extend.h)

На примере функций mouse_on() и mouse_key() становится ясно, как в общем случае должна выглядеть самостоятельно разрабатываемая в C функция для Clipper:

```
<описательная часть модуля>
void <имя функции>
CLIPPER <имя функции>
{
текст программы
}
```

Если Вы теперь готовы приступать к разработке своих собственных функций для Clipper или решились для начала оттранслировать приведенные ниже тексты программ, то помните: при транслировании программы необходимо использовать модель памяти LARGE и отключать контроль проверок стека.

Кроме описанных выше функций mouse_on() и mouse_key(), в листинге 1 приведена функция для вывода строки-подсказки, изменяющейся при нажатии на клавиши Alt, Ctrl или Shift. (Напоминаю, что автор этой программы — Платонов В.П.)

В листинге 2 приведен текст программы на Clipper, демонстрирующей использование функций mouse_on(), mouse_key() и quick_help(). В листинге 3 указано, как скомпоновать исполняемый модуль демонстрационной программы. Библиотека cl.lib поставляется со стандартным пакетом Turbo C.

Приведенные программы можно скомпоновать и на Clipper 5.01 — пример соответствующего командного файла приведен в листинге 4, однако мне неизвестна степень надежности использования функции quick_help() в других программах. Можно лишь посоветовать использовать эту функцию в глубоко вложенных процедурах и не забывать вызывать quick_help() без параметров при выходе из таких процедур. ■


**Литература**

1. Платонов В.П. Alt, Ctrl, Shift //Компьютеры+Программы.– 1992.– № 1.
2. Мизрохи С.В. Turbo Pascal и объектно-ориентированное программирование.– М: Финансы и статистика, 1992.



Монитор–Аспект — 1993

**Подписной купон**

| 1 | 2 | 3 |
|---|---|---|

Монитор Аспект

*Адрес* _____

*Ф.И.О. или название организации* _____

Стоимость подписки на 1 номер журнала "Монитор–Аспект" — 250 рублей. В 1993 году будет выпущено 3 номера (годовая подписка — 750 руб.). Отметьте крестиками те номера журнала, которые Вы желаете получить. Перечислите сумму оплаты на счет НТО "Софт-Москва": р/с 467002 в КБ "Банк развития XXI век", к/с 161816 в РКЦ ГУ ЦБ РФ по г. Москве, МФО 201791. Вышлите подписной купон и квитанцию об оплате по адресу: 117419 Москва, а/я 765.

### Листинг 1. Программа exampl.c для создания объектного модуля exampl.obj

```
/* ---------------------------------------------------------------------
Содержит функции:    mouse_on()  -   включения/выключения "мыши";
                     mouse_key() -   определяет координаты курсора
                                     "мыши" и состояние кнопок;
                     quick_help() -  определяет состояние
                                     клавиш Ctrl, Alt, Shift.
При компиляции установить большую модель памяти и отключить про-
верку на переполнение стека.
При компоновке программы на Clipper использовать exampl.obj и
библиотеку Turbo C - cl.lib:
TLINK <имя программы> .obj+exampl.obj,,,clipper+extend+cl
--------------------------------------------------------------------- */

#include <dos.h>
#include <nandef.h>
#include <extend.h>

/* Описание типов, констант и заголовков */

typedef unsigned short word;

#define StatKeybWord 0x0417-   /* Адрес регистра статуса клавиатуры */
#define Int9_vect 0x09         /* Номер вектора прерываний обработки
                                  нажатия клавиш */
#define VideoMemory (char far*) 0xB8000000L
                               /* Адрес начала видеопамяти */

#define Int33h          0x33L  /* Номер вектора прерываний обработки
                                  действий с "мышью" */

#define LEFT_SHIFT    0x0002
#define RIGHT_SHIFT   0x0001
#define BOTH_SHIFT    0x0003
#define LEFT_CTRL     0x0104
#define RIGHT_CTRL    0x0004
#define LEFT_ALT      0x0208
#define RIGHT_ALT     0x0008

int ini = 0,                   /* Признак инициализации нового
                                  обработчика прерываний
                                  клавиатуры */
    str,
    rez;

char *video = VideoMemory;

char i = 31;

char *Nothing,
     *Shift,
     *Ctrl,
     *Alt;

void (interrupt far *old_Int9)(void);   /* Указатель на старое значение
                                           вектора обработки прерываний от
                                           клавиатуры */
word *key = (word far*)StatKeybWord; /* Указатель на регистр
                                        состояния клавиатуры */

int mouse_init();    /* Функция проверки наличия драйвера "мыши" */
void mouse_show();/* Функция делает видимым курсор "мыши" на
                     экране */
void mouse_hide(); /* Функция делает невидимым курсор "мыши" на
                      экране */

void interrupt far new_Int9(void);      /* Заголовок новой функции
                                           обработки прерываний от
                                           клавиатуры */
void write_screen();          /* Функция вывода сообщения на экран */
void say_screen();            /* Функция анализа нажатия клавиш */

CLIPPER mouse_on()
{
    if (PCOUNT = = 1)
        {ini = _parni(1);
        if (mouse_init() > 0)
            if (ini)
                {mouse_show();
                ini = 1;
                }
        else
            {mouse_hide();
            ini = 0;
            }
        }
    else
        {mouse_hide();
        ini = 0;
        }
    _retni(ini);
}

/* Функция проверки наличия драйвера "мыши". Возвращает 0, если
"мыши" нет, и количество кнопок в противном случае */

int mouse_init()

{
struct REGPACK Reg;

Reg.r_ax = 0;
intr(Int33h, &Reg);
return Reg.r_bx;
}

/* Функция делает видимым курсор "мыши" */

void mouse_show()

{
struct REGPACK Reg;

Reg.r_ax = 1;
intr(Int33h, &Reg);
}

/* Функция делает невидимым курсор "мыши" */

void mouse_hide()

{
struct REGPACK Reg;

Reg.r_ax = 2;
intr(Int33h, &Reg);
}

/* Функция определяет, нажата ли кнопка, и координаты курсора
"мыши" */

CLIPPER mouse_key()
{
struct REGPACK Reg;
int key, x, y;

Reg.r_ax = 3;
intr(Int33h, &Reg);
key = Reg.r_bx;
x = Reg.r_cx/8;
y = Reg.r_dx/8;
rez = 1000*y+10*x+key;
_retni(rez);
}

/* Функция изменения обработки прерываний от клавиатуры для
отслеживания нажатия на клавиши Ctrl, Alt, Shift */

CLIPPER quick_help()
{
```

```
    if (PCOUNT = = 5)
        {str = _parnl(1);
        Nothing = _parc(2);
        Shift = _parc(3);
        Ctrl = _parc(4);
        Alt = _parc(5);

        say_screen();

        if (ini = = 0)
            {ini = 1;
            old_Int9 = getvect(Int9_vect);
            setvect(Int9_vect,new_Int9);
            }
        else
            ini = 2;
        }
    else
        if (ini ! = 0)
            {setvect(Int9_vect,old_Int9);
            ini = 0;
            }
    _retni(ini);
}

void interrupt new_Int9()
{
(*old_Int9) ();
say_screen();
}

void write_screen(char far *num)
{
char far *t;
char l;

t = video + (160*str);
l = 0;
while (80-l++)
    {*t++ = *num++;
    t++;
    }
}

void say_screen()
{
word status = *key;
status & = 0x000f;
switch(status)
    {case LEFT_SHIFT:
    case RIGHT_SHIFT:
    case BOTH_SHIFT :
        write_screen(Shift);
        break;
    case LEFT_CTRL   :
    case RIGHT_CTRL :
        write_screen(Ctrl);
        break;
    case LEFT_ALT    :
    case RIGHT_ALT   :
        write_screen(Alt);
        break;
    default          :
        write_screen(Nothing);
    }
}
```

**Листинг 2. Пример программы на Clipper–87 ex_clip.prg — использование дополнительных функций mouse_on(), mouse_key() и quick_help()**

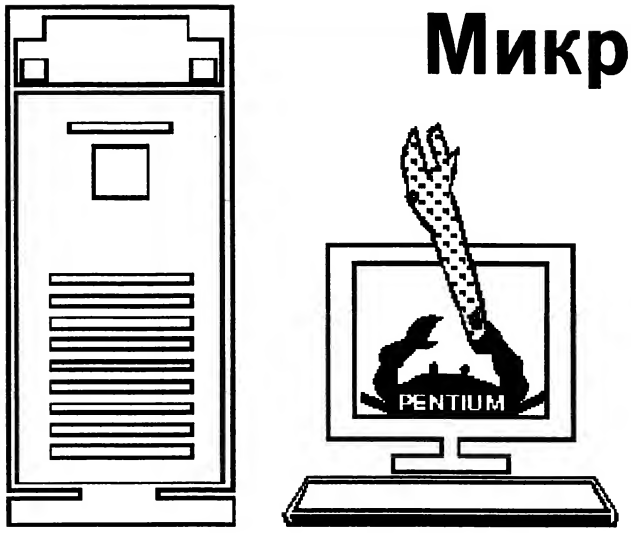
```
PRIVATE s_h, Nothing, Shift, Ctrl, Alt, pp
CLEAR SCREEN
SET CURSOR OFF
S_H = 24
Nothing  = '1 Help 2 Menu  3 Wind2  4 Copy   5'+SPACE(38)+'10 QUIT'
Shift    = '1 ...  2       3        4 Del    5'+SPACE(38)+'10 ...'
Ctrl     = '1 ...  2       3        4 Ren    5'+SPACE(38)+'10 !!!'
Alt      = '1 ...  2       3        4 Move   5'+SPACE(38)+'10 ???'

SET COLOR TO 'N/BG'
FOR l = 0 TO 9
    @ 24, 2+l*8 SAY ' '
NEXT

quick_help(S_H,@Nothing,@Shift,@Ctrl,@Alt)

SET COLOR TO 'W/N'
mouse_on(1)
@ 0, 6 SAY 'Пример программы обработки состояния клавиш Shift, Ctrl,
Alt и "мыши"'
@ 1, 20 SAY 'Для окончания работы нажмите любую клавишу'
SET COLOR TO 'RB/N'
@ 5, 10 SAY 'Вертикальная координата курсора "мыши":'
@ 7, 10 SAY 'Горизонтальная координата курсора "мыши":'
@ 9, 10 SAY 'Состояние клавиш "мыши":'
SET COLOR TO 'W+/N'
ini = 1
DO WHILE .T.
    key = mouse_key() && Считываем состояние клавиш и курсора "мыши"
    y = INT(key/1000)
    x = INT((key-1000*y)/10)
    key = key-1000*y-10*x
    @ 5, 51 SAY y PICTURE '99'
    @ 7, 51 SAY x PICTURE '99'
    DO CASE
        CASE key = 0
            IF ini < > 0
            @ 9, 51 SAY 'Кнопки не нажаты '
            ini = 0
            ENDIF
        CASE key = 1
            @ 9, 51 SAY 'Нажата левая кнопка '
            ini = 1
        CASE key = 2
            @ 9, 51 SAY 'Нажата правая кнопка '
            ini = 1
        CASE key = 4
            @ 9, 51 SAY 'Нажата средняя кнопка'
            ini = 1
    ENDCASE
    IF INKEY(0.01) < > 0
        EXIT
    ENDIF
ENDDO

quick_help()
mouse_on(0)
SET COLOR TO 'W/N'
CLEAR

QUIT
```

**Листинг 3. Пример командного файла comp.bat для компоновки исполнительного модуля ex_clip.exe программы ex_clip.prg (см. листинг 2)**

```
CLIPPER ex_clip.prg
IF ERRORLEVEL 1 EXIT
TLINK ex_clip+exampl,,,clipper+extend+cl
```

**Листинг 4. Пример командного файла comp.bat для компоновки исполнительного модуля ex_clip.exe программы ex_clip.prg (см. листинг 2) с использованием Clipper5.01**

```
clipper ex_clip.prg
IF ERRORLEVEL 1 EXIT
TLINK ex_clip+exampl,,,clipper+extend+terminal+cl
```




# Микропроцессор Pentium

*Д. Бройтман*

## Часть 2
## Программирование — основная модель

Основная модель описывает среду программирования в Pentium с точки зрения программиста на ассемблере.

Она включает в себя:
- организацию памяти;
- типы данных;
- регистры;
- формат команд;
- выборку операндов;
- прерывания и исключительные ситуации.

### 1. Организация памяти

Память, расположенная на шине Pentium, называется физической памятью. Она организована в виде последовательности 8-битовых байтов. Каждому байту присвоен уникальный адрес — физический адрес, который изменяется от 0 до $2^{32}$-1 (4 гигабайта).

**Управление памятью** — это аппаратный механизм, организующий использование памяти. При использовании управления памятью программы обращаются не непосредственно к физической памяти, а к модели памяти, называемой виртуальной памятью.

Управление памятью включает сегментацию и управление страницами.

**Сегментация** поддерживает несколько независимых адресных пространств. Управление страницами поддерживает модель большого адресного пространства в ОП, используя как ОП, так и дисковую память. Адрес, выдаваемый программой, называется логическим адресом. Аппаратура сегментации переводит логический адрес в адрес в непрерывном адресном пространстве, называемый линейным адресом. Аппаратура управления страницами переводит линейный адрес в физический.

Память может рассматриваться как единое адресное пространство либо как набор независимых адресных пространств, называемых сегментами. Одна программа может иметь до 16383 сегментов различных размеров и типов. Разделение на сегменты увеличивает надежность, например, отдельные сегменты могут быть назначены для кода, данных и стека.

Вне зависимости от того, один или несколько сегментов используются, логический адрес рассматривается как смещение внутри сегмента. У каждого сегмента есть дескриптор, содержащий базовый адрес сегмента и его предельный размер. Если смещение не превосходит этого предела, то адрес получается сложением базового адреса сегмента со смещением в логическом адресе.

Бит 31 регистра CR0 указывает, включено ли управление страницами. Если этот бит установлен в 1, то перевод линейного адреса в физический осуществляет аппаратура управления страницами. В противном случае линейный адрес совпадает с физическим.

**Управление страницами** разбивает линейное адресное пространство на участки, называемые страницами, которые могут находиться либо в ОП, либо на диске. Логический адрес либо переводится в адрес на странице в ОП, либо генерируется исключительная ситуация, при обработке которой операционная система читает страницу с диска в память и обновляет карту страниц.

Разбиение на сегменты является видимым для прикладного программиста; управление страницами невидимо для него и далее обсуждаться не будет.

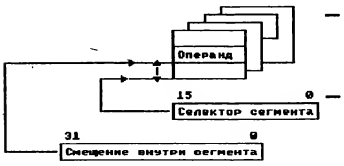
#### 1.1. Плоская (бессегментая) модель памяти

Это простейшая модель памяти, когда всем сегментам соответствует одно и то же пространство физической памяти.

### 1.2. Сегментная модель

Логическое адресное пространство состоит из 16383 ($2^{14}$-1) сегментов, каждый размером до 4 гигабайтов ($2^{32}$). Процессор устанавливает соответствие между логическим адресным пространством размером $2^{46}$ байтов (64 терабайта) и физическим адресным пространством с помощью механизмов трансляции адреса. Детали этой трансляции для прикладного программиста не важны.

Указатель на сегментное адресное пространство состоит из двух частей (рис. 1):




- 16-битовый селектор сегмента, определяющий сегмент;
- 32-битовое смещение внутри сегмента.

Процессор использует селектор сегмента, чтобы найти линейный адрес начала сегмента, названный базовым адресом. Программа обращается к памяти, используя фиксированные смещения от базового адреса, так что объектный модуль может загружаться в разные сегменты без изменения адресов внутри него. Размер сегмента определяется программистом и может в точности совпадать с размером загруженного в него модуля.

**Рис. 1. Указатель на сегментное адресное пространство**

## 2. Типы данных

Фундаментальные типы данных представлены на рис. 2. Это байты, слова, двойные слова и квадрослова.

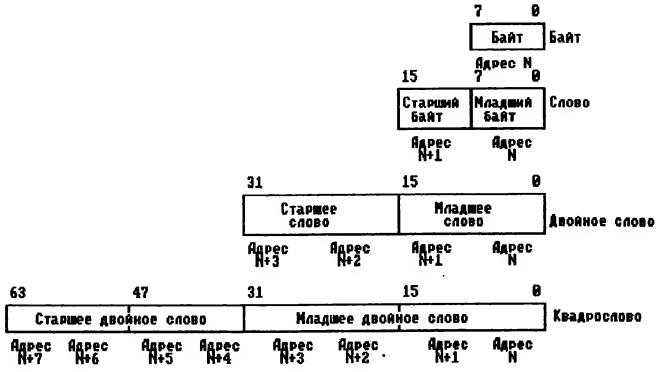



**Рис. 2. Фундаментальные типы данных**

Каждый из представленных на рисунке 2 типов данных может начинаться с любого адреса: это означает, что слово не обязано начинаться с четного адреса; двойное слово — с адреса, кратного 4, квадрослово — с адреса, кратного 8. Таким образом достигается максимальная гибкость структур данных и эффективность использования памяти. Однако обмен данными между процессором и памятью осуществляется в Pentium через 64-битовую шину данных и для достижения максимальной производительности этого обмена желательно выравнивать слова по четным адресам, двойные слова — по адресам, кратным 4, и квадрослова — по адресам, кратным 8.

На основе фундаментальных типов данных строятся все остальные типы данных, распознаваемые ко-

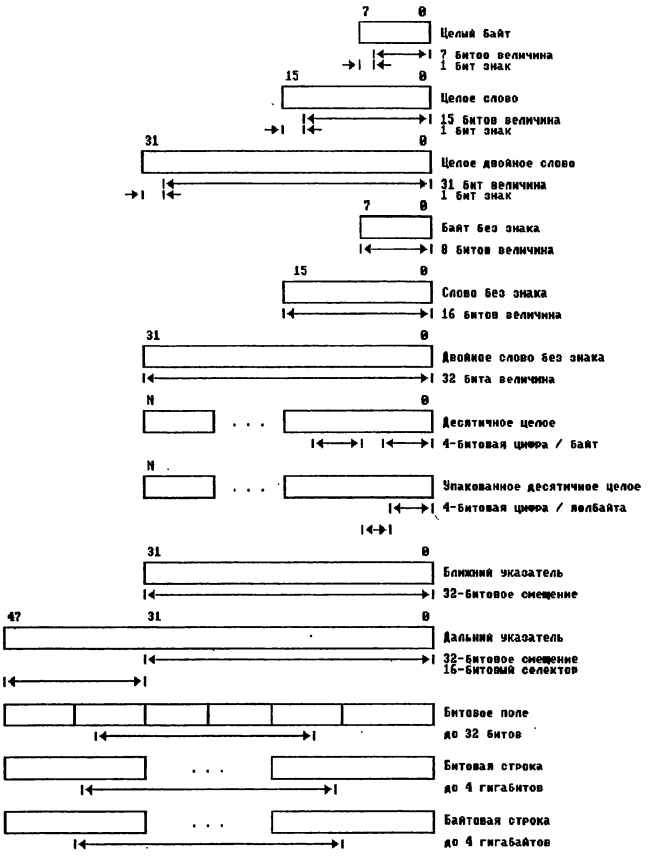



**Рис. 3. Типы данных Pentium**

мандами процессора. Все они, за исключением типов данных с плавающей точкой, представлены на рис. 3.

1) Целое (Integer). Двойное число со знаком, занимающее байт, слово или двойное слово и представленное в дополнительном коде.

2) Порядковое, или число без знака (Ordinal). Неотрицательное двойное число, занимающее байт, слово или двойное слово.

3) Десятичное целое (BCD Integer). Двоично-кодируемая запись десятичного числа; каждая цифра занимает 1 байт; ее величина — двойное значение младшего полубайта. Старший полубайт должен быть равен 0 при умножении и делении; может содержать что угодно при сложении и вычитании.

4) Упакованное десятичное целое (Packed BCD Integer). Двоично-кодируемая запись десятичного числа: каждая цифра занимает полбайта; по две цифры в байте.

5) Ближний указатель (Near Pointer). 32-битовое смещение внутри сегмента.

6) Дальний указатель (Far Pointer). 48-битовый логический адрес, состоящий из 16-битового селектора сегмента и 32-битового смещения.

7) Битовое поле (Bit field). Непрерывная последовательность битов, начинающаяся в любой позиции любого байта и содержащая до 32 битов.

8) Битовая строка (Bit string). Непрерывная последовательность битов, начинающаяся в любой позиции любого байта и содержащая до $2^{32}$-1 байтов.

9) Байтовая строка. Непрерывная последовательность байтов, слов, двойных слов, которая может содержать от 0 до $2^{32}$-1 байтов.

## 3. Регистры

Процессор содержит 16 регистров, которые могут использоваться прикладным программистом (рис. 4): 8 регистров общего назначения, 6 сегментных регистров и два регистра управления и состояния процессора.

**Рис. 4. Регистры Pentium**

### 3.1 Регистры общего назначения

Регистры общего назначения — это 32-битовые регистры EAX, EBX, ECX, EDX, EBP, ESP, ESI и EDI. Некоторые из входящих в них 16-битовых слов и байтов также имеют собственные имена (рис. 4), по которым к ним можно обращаться. Эти регистры могут содержать операнды для логических и арифметических операций и для вычисления адреса (регистр ESP не может быть индексным операндом). Во все эти регистры могут также помещаться результаты логических, арифметических операций.

Некоторые команды используют заранее оговоренные регистры в качестве операндов и для записи результатов. Это команды умножения и деления с двойной точностью, ввода-вывода, строковой перекодировки (translate), цикла (loop), сдвига и вращения переменной длины, операции над стеком.

### 3.2. Сегментные регистры

Сегментные регистры CS, DS, SS, ES, FS и GS содержат 16-битовые селекторы сегментов, которые выступают как индексы в таблицах, содержащих битовый адрес каждого сегмента и другую информацию, относящуюся к доступу к памяти. Таким образом, шесть сегментов памяти доступны из программы немедленно, для доступа в другом сегменте необходимо загрузить их селекторы в сегментные регистры.

Сегмент, содержащий выполняемые команды, называется кодовым сегментом. Его селектор содержится в регистре CS. Процессор выбирает команды из кодового сегмента, используя содержимое регистра EIP как смещение внутри сегмента. Регистр CS загружается в результате прерываний, исключительных ситуаций и выполнения команд передачи управления между сегментами.

Перед вызовом процедуры выделяется участок памяти под стек. Стек содержит адрес возврата, передаваемые процедуре параметры и временные переменные, которые будет создавать процедура. Все операции со стеком используют для поиска стекового сегмента регистр SS. Он может быть загружен явно в прикладной программе.

Четыре регистра для сегментов данных (DS, ES, FS, GS) обеспечивают эффективное и безопасное разделение доступа к различным видам данных.

### 3.3. Организация стека

Организация стека представлена на рис. 5. Операции со стеком используют регистры SS, ESP и EBP.

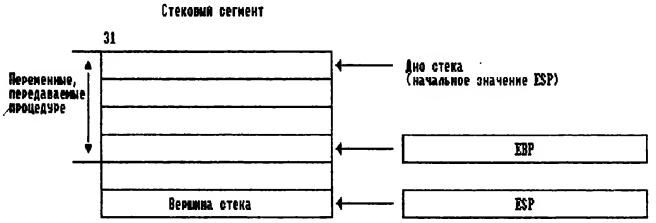



**Рис. 5. Организация стека**

**1. Регистр стекового сегмента (SS).** Стеки находятся в памяти. Число стеков ограничено максимальным числом сегментов; максимальный размер стека — 4 гигабайта (так же, как и у сегмента). В каждый момент времени доступен только один стек — тот, чей селектор сегмента находится в регистре SS. Он называется текущим стеком. Процессор автоматически использует во всех операциях со стеком регистр SS.

**2. Указатель стека (регистр ESP).** Содержит смещение вершины стека в текущем стековом сегменте. При вталкивании объекта в стек регистр ESP уменьшается; при выталкивании — увеличивается. Таким образом, стек растет вниз по направлению к меньшим адресам памяти.

**3. Базовый указатель рамки стека (EBP).** Обычно используется для доступа к структурам данных, передаваемым в стек. При вызове процедуры стек содержит адрес возврата и передаваемые процедуре структуры данных. Если в этот момент сохранить ESP в регистре EBP, то с помощью EBP можно будет обращаться к этим структурам данных с фиксированным смещением. В противном случае смещение , отсчитываемое от ESP, будет постоянно меняться при создании и ликвидации временных переменных в процедуре.

### 3.4. Регистр флагов

Регистр флагов EFLAGS содержит флаги (то есть биты), управляющие определенными операциями и показывающие состояние процессора. Этот регистр содержит и коды условия (то есть перенос, знак, переполнение и тому подобное), устанавливаемые в результате выполнения арифметических и логических команд и анализируемые командами перехода.

### 3.5. Указатель команд

Регистр — указатель команд EIP содержит смещение в кодовом сегменте следующей команды, подлежащей

выполнению. Он не доступен явно для программиста, а изменяется командами передачи управления, прерываниями и исключительными ситуациями.

## 4. Формат команд

Команда Pentium может включать следующие составные части.

1) Префикс — один или более байтов, предшествующих команде и изменяющих порядок ее выполнения.

а) Перекрытие сегмента — префикс явно указывает, какой сегментный регистр должна использовать команда.

б) Размер адреса — переключение между 16-битовой и 32-битовой адресацией.

в) Размер операнда — переключение между 16-битовым и 32-битовым размером данных.

г) Повторение — строковая команда должна быть повторена для каждого элемента строки.

д) Закрыть (lock) — для исключительного использования раздельной памяти в мультипроцессорном окружении.

2) Код операции. Для некоторых операций существует несколько кодов, по одному для каждой разновидности операций.

3) Спецификатор регистра указывает, что в команде присутствуют один или два регистровых операнда.

4) Спецификатор режима адресации определяет, является ли операнд регистром или ячейкой памяти; если в памяти, то будут ли использованы смещения, базовый регистр, индексный регистр, масштабирование.

5) Байт с базовым регистром, индексным регистром и масштабирующим фактором.

6) Смещение — целое со знаком размером 32, 16 или 8 битов.

7) Непосредственный операнд. Байт, слово или двойное слово, непосредственно содержащее значение операнда.

## 5. Выборка операндов

У команды может не быть операндов (например, NOP), а может быть один или более операндов. Операнд может находиться в самой команде (непосредственный), в регистре, в печати, в карте ввода-вывода.

В команде могут присутствовать явные и неявные операнды. Неявные операнды есть в большинстве команд, так, например, все арифметические команды изменяют регистр EFLAGS.

Регистровые и непосредственные операнды находятся на шине процессора. Непосредственный операнд оказывается там в ходе предвыборки до начала выполнения команды.

Выборка операнда из памяти включает выборку сегмента и вычисление адреса.

### 5.1. Выборка сегмента

Сегмент может быть задан в префиксе перед кодом команды. В противном случае используются следующие соглашения.

1) Для выборки команд используется кодовый сегмент (регистр CS).

2) Операции над стеком используют стековый сегмент (регистр SS).

3) При операциях над всеми данными, кроме относящихся к стеку и результатов строковых команд, используется сегмент данных (регистр DS).

4) Операнды — результаты выполнения строковых команд размещаются в сегменте, задаваемом регистром ES.

Префиксы могут задавать любой сегментный регистр. Однако префиксы не действуют в следующих случаях.

1) Результаты строковых операций — всегда в сегменте ES.

2) Результат выталкивания и источник вталкивания в стек используют всегда регистр SS.

3) Выборка команд всегда использует регистр CS.

### 5.2. Вычисление адреса

Если операнд находится в памяти, то смещение его внутри сегмента определяется четырьмя компонентами: смещением, базовым регистром, индексным регистром и масштабирующим фактором. Программист выбирает тот или иной способ вычисления адреса, в зависимости от характера обрабатываемых данных. В наиболее общем случае адрес вычисляется по формуле:

```
Segment + base + (index * scale) + displacement
Здесь segment      — адрес сегмента,
     base          — содержимое базового регистра,
     index         — содержимое индексного регистра,
     scale         — масштабирующий фактор (1, 2, 4, 8),
     displacement  — смещение.
```

## 6. Прерывания и исключительные ситуации

У процессора есть два механизма для приостановки выполнения программ.

**1) Исключительные ситуации** — синхронные события, являющиеся ответом процессора на определенные условия, возникшие в ходе выполнения команды.

**2) Прерывания** — асинхронные события, как правило, являющиеся результатом воздействия внешних устройств.

Прерывания и исключительные ситуации похожи в том, что и те и другие вынуждают процессор временно приостановить выполнение текущей программы и запустить программу с более высоким приоритетом.

Главное различие между прерываниями и исключительными ситуациями следует из их происхождения. Исключительная ситуация всегда воспроизводится при повторном выполнении вызвавшей ее программы; взаимодействие прерывания с программой не является таким однозначным.

Прикладные программисты, как правило, не имеют дела с обработкой исключительных ситуаций и прерываний. Однако определенные виды исключительных ситуаций важны для прикладного программирования, и операционные системы, как правило, предоставляют интерфейс между прикладной программой и механизмом обработки исключительных ситуаций в процессоре.

*Автор выражает благодарность сотрудникам московского представительства Intel Corporation за содействие при подготовке статьи.* ■

# Создание демонстрационных программ в MS-DOS

*В. Плотников*

*Автор благодарит сотрудника ВЦ космодрома «Плесецк» Евстратова И.В. за помощь, оказанную при написании статьи.*

Создание программного продукта — процесс длительный и сложный, и в конце него возникает необходимость найти покупателей для разработанных программ. Наиболее удобно это осуществить с помощью рекламы, а одним из ее видов является распространение демонстрационных вариантов созданного программного обеспечения.

Основной принцип работы демонстрационной программы заключается в том, что она показывает функциональные возможности реального программного продукта. Причем это осуществляется без внешних воздействий со стороны оператора (таких как нажатие определенных клавиш для управления ходом выполнения программы). Таким образом, демонстрационная программа должна имитировать работу оператора с клавиатурой. Клавиатуру в операционной системе (ОС) MS-DOS обслуживают прерывания с номерами 09h, 16h, а также отдельные функции прерывания 21h. Эмуляцию нажатий клавиш в демонстрационном примере можно осуществить заменой системных обработчиков этих прерываний на собственные. Поэтому коротко рассмотрим принципы работы обработчиков прерываний, обслуживающих клавиатуру (более подробное описание приведено в [1]).

Аппаратное прерывание 09h возникает всякий раз, когда происходит нажатие или отпускание клавиши. Оно предназначено для чтения скан-кода нажатой клавиши, получения соответствующего ASCII кода и записи этих значений в буфер клавиатуры, который представляет собой участок оперативной памяти (ОП), адресуемый относительно сегментного адреса 0040h. BIOS содержит характеристики этого буфера, так в словах ОП содержатся смещения его начала (по адресу 0040h:0080h) и конца (по адресу 0040h:0082h) относительно сегмента 0040h (на машинах IBM PC обычно буфер располагается в памяти с адреса 0040h:001Eh и занимает область длиной 32 байта). Указатель на начало информации в буфере, помещенной туда при вводе с клавиатуры, располагается по адресу 0040h:001Ah; указатель на конец информации в буфере имеет адрес 0040h:001Ch (если содержимое этих ячеек памяти совпадает, то буфер клавиатуры пуст).

Прерывание 16h предназначено для программного обслуживания буфера клавиатуры компьютера. Функции 00h (чтение символа с ожиданием) или 10h (чтение символа с ожиданием для 101-клавишной клавиатуры) позволяют извлечь скан-код и код символа из этого буфера. Другие функции предназначены для проверки наличия символа в буфере, записи символа в буфер, чтения характеристик переключающих клавиш и установки временных характеристик работы клавиатуры.

Операционная система MS-DOS предоставляет средства работы с клавиатурой в виде следующих функций прерывания 21h:

- чтение символа с клавиатуры и эхо-вывод (функция 00h);
- чтение символа из клавиатурного буфера (функция 06h);
- чтение символа с клавиатуры без эхо-вывода (функция 07h);
- нефильтрованное чтение символа с клавиатуры без эхо-вывода (функция 07h);
- ввод строки символов с клавиатуры (функция 0Ah);
- проверка состояния клавиатуры (функция 0Bh);
- сброс буфера клавиатуры и чтение символа (функция 0Ch).

Эти функции рассматриваться в статье не будут, так как они для получения скан-кода и символа нажатой клавиши обращаются к прерыванию BIOS с номером 16h. Единственной особенностью их работы является то, что если клавиатурный буфер пуст, то они переводят задачу в состояние ожидания до момента возникновения аппаратного прерывания от клавиатуры (то есть

до заполнения буфера клавиатуры). Для обеспечения другой системной деятельности в случае отсутствия кодов символов в буфере клавиатуры эти функции порождают прерывание 28h. Программа его обработки в MS-DOS состоит из одного оператора IRET. Поэтому программисту предоставляется возможность написать собственный обработчик указанного прерывания, который будет осуществлять определенные действия при ожидании ввода с клавиатуры через функции MS-DOS.

Необходимо отметить, что если программа написана на языке высокого уровня (C, Pascal и так далее), то операторы этих языков, реализующие ввод с клавиатуры, также используют рассмотренные прерывания. Поэтому для любой программы, выполняющейся в ОС MS-DOS, при разработке демонстрационного примера имитация нажатий клавиш должна осуществляться (на низком уровне):

- при вызове прерывания 16h для чтения символа из клавиатурного буфера;
- при ожидании ввода с клавиатуры функциями прерывания 21h (то есть при инициировании ОС прерывания 28h).

Таким образом, для демонстрационной программы надо определить новые обработчики этих прерываний, которые бы работали не с буфером клавиатуры, а с областью памяти программы пользователя, отведенной для хранения кодов ASCII и скан-кодов клавиш (нажатие которых надо имитировать). С целью внесения минимальных изменений в имеющийся программный продукт рекомендуется создать отдельную процедуру (подпрограмму), выполняющую имитацию обмена с клавиатурой. Алгоритм ее работы будет заключаться в выполнении следующих действий (шагов).

1. Сохранение значений регистров, используемых в подпрограмме (это необходимо для тех модифицируемых программ, которые анализируют или используют в процессе своего выполнения начальные установки отдельных регистров ОС).

2. Запрещение аппаратных прерываний от клавиатуры.

3. Запоминание системных адресов обработчиков прерываний обслуживания клавиатуры.

4. Замена векторов прерываний 16h и 28h на новые, которые находятся в теле процедуры и предназначены для имитации обслуживания клавиатуры.

5. Чистка буфера клавиатуры с целью исключения кодов клавиш, которые были случайно нажаты в момент вызова программы.

6. Восстановление значений регистров и передача управления основной программе.

Новый обработчик прерывания 28h предназначен для обслуживания системного буфера клавиатуры. При возникновении данного прерывания происходит проверка наличия кода символа в клавиатурном буфере. Если буфер не пуст, то обработка прерывания прекращается. При отсутствии там информации в буфер клавиатуры помещается код текущего символа из области процедуры, где содержатся сведения о клавишах, нажатие которых надо имитировать. После посылки туда символа из определенных в области процедуры кода ASCII и скан-кода восстанавливаются векторы прерываний 16h и 28h, разрешаются аппаратные прерывания от клавиатуры.

Прерывание BIOS 16h также извлекает коды из буфера клавиатуры (это осуществляется его инициированием с функциями 00h и 10h из прикладных программ). Новый обработчик данного прерывания при отработке этих функций будет вызывать прерывание 28h для заполнения буфера, а затем передавать управление по адресу старого прерывания BIOS обслуживания клавиатуры. В случае вызова прикладными программами рассматриваемого прерывания с другими значениями функций его обслуживание должно производиться также старым обработчиком прерывания 16h.

В листинге 1 приведен текст простейшей демонстрационной программы на языке ассемблера IBM PC. Эта программа предназначена для вывода на экран терминала текстового сообщения после нажатия на любую клавишу. Убрав из текста программы оператор CALL DEMO, получим из демонстрационной программы обыкновенную, то есть реально функционирующую задачу, требующую действий оператора для ввода информации с клавиатуры. Внутренняя процедура DEMO предназначена для эмуляции воздействий клавиатуры. Каждый из читателей может использовать ее для модификации своих программ на ассемблере, внеся изменения только в область данных этой процедуры, где размещены коды имитируемых клавиш. При использовании подпрограммы DEMO для доработки программ, написанных на языках высокого уровня, в целях создания демонстрационного варианта в ее текст необходимо внести изменения, предусмотренные соглашениями о связях между ассемблером и конкретным языком программирования.

Предлагаемый метод модификации программ описывает один из простейших способов создания демонстрационных задач. Следовательно, чтобы при доработке программных продуктов не зависеть от требований конкретного языка программирования, предлагается оформить процедуру превращения программы в демонстрационный вариант (подобную описанной в данной статье подпрограмме DEMO) в виде вируса. Тогда работы по созданию демонстрационных примеров можно вести не на уровне исходных текстов, а на уровне исполняемых модулей. Кроме того, в более сложном варианте такой процедуры рекомендуется хранить данные об имитируемых нажатиях клавиш, допустим, в конце файла, содержащего загрузочный модуль задачи. Процедура такого превращения должна в момент вызова считать информацию о кодах ASCII и скан-кодах клавиш из файла в свою область данных. ■

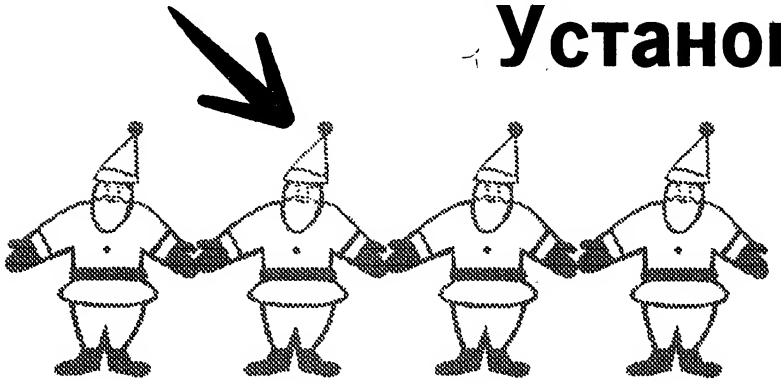
**Литература**

1. Фролов А.В., Фролов Г.В. Аппаратное обеспечение IBM PC.– М.: Диалог-МИФИ, 1992.

## Листинг 1. Текст демонстрационного варианта программы на языке ассемблера

```
        title demo
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Программа отображения на экране терминала текстового
;     сообщения после нажатия пользователем любой
;           клавиши на клавиатуре компьютера
;                 (демонстрационный вариант)
;
;------------------------------------------------------------------
;
; Используемая процедура:
;
; DEMO  -     внутренняя (near) процедура организации ими-
;             тации воздействий оператора на
;             ход выполнения программы через клавиатуру
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

data      segment  'DATA'
msg1      db   'Н а ж м и т е   к л а в и ш у'
          db   0Dh,0Ah,0Dh,0Ah,'$'
msg2      db   ' Тестовая программа работы демострационного примера$'
data      ends

stack     segment  'stack'
          dw   60 dup (?)
stack     ends

          assume    cs:code,ss:stak

code      segment

demonstr proc       far
;
; Вызов процедуры превращения обычной программы в
;     демонстрационную
;

          call  demo

;
; Начало реальной программы
;

          mov       ax,data
          mov       ds,ax
          assume    ds:data

;
; Вывод указания оператору на экран терминала
;

          mov ah,9
          lea  dx,msg1
          Int   21h

;
; Ожидание нажатия оператором клавиши
;

          xor  ax,ax
          int   16h
;
; Вывод текстового сообщения на экран терминала
;

          mov ah,9
          lea  dx,msg2
          Int   21h

;
; Завершение работы программы
;

          mov ax,4C00h
          Int   21h

demonstr endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;     Процедура организации имитации
; воздействий оператора на ход выполнения
;     программы через клавиатуру
;
;-------------------------------------------------------
;
; Процедура сохраняет необходимые для ее работы регист-
; ры, а также значения флагов при входе в нее. На время
; эмуляции операций с клавиатурой подменяет системные
; векторы прерываний 16h и 28h на собственные.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
demo      proc      near
;
; Сохранение значений сегментных, рабочих регистров
;           и флагов в стеке
;
          push ax
          push es
          pushf
;
; Обнулим счетчик обработанных клавиш
;

          mov cs:count_key,0

;
; Запретим прерывания от клавиатуры
;

          In   al,61h     ; Здесь происходит блокирование
          and  al,7Fh     ; клавиатуры, поэтому нажатие
          out  61h,al     ; любых клавиш оператором
                          ; не будет восприниматься ОС
;
; Сохранение адресов системных
;     прерываний 16h и 28h
;
          xor  ax,ax
          mov es,ax

          mov ax,word ptr es:[16h*4]
          mov cs:Int16_old,ax
          mov ax,word ptr es:[16h*4+2]
          mov cs:Int16_old+2,ax

          mov ax,word ptr es:[28h*4]
          mov cs:int28_old,ax
          mov ax,word ptr es:[28h*4+2]
          mov cs:int28_old+2,ax
;
; Подмена системных векторов прерываний
;     обслуживания клавиатуры (16h и 28h)
;

          cli
          mov ax,cs

          mov word ptr es:[16h*4],offset cs:Int16_new
          mov word ptr es:[16h*4+2],ax

          mov word ptr es:[28h*4],offset cs:Int28_new
          mov word ptr es:[28h*4+2],ax

          sti

;
; Очистка буфера клавиатуры
;

          mov al,byte ptr es:[480h]
          mov byte ptr es:[41Ah],al
          mov byte ptr es:[41Ch],al

;
; Восстановление значений сегментных, рабочих регистров
;           и флагов в стеке
;

          popf
          pop  es
          pop  ax

;
; Завершение работы процедуры
;

          ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;           Область данных процедуры DEMO
;
;--------------------------------------------------------
;
int28_old     dw   0     ; Область для хранения
              dw   0     ; системного адреса прерывания 28h
count_key     dw   ?     ; Текущее число клавиш, нажатие
                         ; которых было эмулировано
```

```
add_key         dw   1      ; Поле определяет число клавиш,
                            ; нажатие которых будет имити-
                            ; роваться в процессе выполнения
                            ; демонстрационной программы

code_key        labelword; Метка определяет смещение
                            ; области данных, в которой
                            ; находятся коды эмулируемых
                            ; клавиш

        db   0Dh            ; ASCII код  <Enter>
        db   1Ch            ; Скан-код  <Enter>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;      Процедура обслуживания прерывания 28h
;
;------------------------------------------------------------
;
; Процедура сохраняет значения необходимых для ее
; выполнения регистров. Если буфер клавиатуры пуст, по-
; мещает туда коды эмулируемой клавиши. После отправки
; в буфер сведений о последней имитации нажатия кла-
; виши происходит восстановление системных векторов
; прерываний 16h и 28h, а также разблокирование клавиа-
; туры.
;
; Данный обработчик прерывания вызывается из нового
; обработчика int 16h или порождается ОС во время ожидания
; ввода с клавиатуры, реализованного через функции DOS.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
int28_new       label       near

        push    ax
        push    si
        push    bx
        push    es

;
; Проверка заполнения буфера клавиатуры
;

        xor  ax,ax
        mov  es,ax
        mov  al,byte ptr es:[41Ah]
        cmp  al,byte ptr es:[41Ch]
        jz   int28_1

        jmp  int28_2                ; Есть символ в буфере

;
; Помещаем символ в буфер клавиатуры
;
int28_1:

        sti

        mov cx,30
int28_4:
        push cx
        xor  cx,cx
int28_3:            ;
        loop int28_3                ; Пауза для увеличения
                                    ; интервалов времени между
                                    ; имитацией нажатий клавиш

        pop  cx
        loop int28_4

        cli
        mov  bx,word ptr es:[480h]  ; Очистка буфера
        mov  byte ptr es:[41Ah],bl  ; клавиатуры
        mov  byte ptr es:[41Ch],bl  ; компьютера

        mov  ax,40h
        mov  es,ax

        mov  si,cs:count_key
        shl  si,1
        mov  ax,cs:code_key[si]
        mov  word ptr es:[bx],ax    ; Помещение в буфер
        add  byte ptr es:[1Ch],2    ; клавиатуры кода
                                    ; имитируемой
                                    ; клавиши

        inc  cs:count_key
        mov  ax,cs:count_key

        cmp  ax,cs:add_key          ; В буфер поместили
                                    ; код
                                    ; последней клавиши?

        jnz  int28_2                ; Нет

;
; Заключительные операции по окончанию
; процесса имитации нажатий клавиш
;

        xor  ax,ax
        mov  es,ax

        mov ax,cs:int16_old                 ; Восстановление
        mov word ptr es:[16h*4],ax          ; системного
        mov ax,cs:int16_old+2               ; адреса
        mov word ptr es:[16h*4+2],ax        ; int 16h

        mov ax,cs:int28_old                 ; Восстановление
        mov word ptr es:[28h*4],ax          ; системного
        mov ax,cs:int28_old+2               ; адреса
        mov word ptr es:[28h*4+2],ax        ; int 28h

        in   al,61h
        or   al,80h
        out  61h,al                         ; Разблокирование клавиатуры

;
; Выход из процедуры
;
int28_2:

        pop  es
        pop  bx
        pop  si
        pop  ax

        sti

        iret


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;      Процедура обработки прерывания 16h
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
int16_new       label       near
;
;      Проверка вызова данного прерывания
; прикладной программой для чтения символа
;      с ожиданием (функция 00h)
;

        or   ah,ah
        jz   int16_1

;
;      Проверка вызова данного прерывания
; прикладной программой для чтения символа
; с ожиданием для 101-клавишной клавиатуры
;           (функция 10h)
;

        cmp ah,10h
        jnz  int16_2

;
; Заполнение буфера клавиатуры информацией об
; имитации очередного нажатия клавиши
;
int16_1:

        int  28h

;
; Передача управления системному прерыванию
;      обслуживания клавиатуры
;
int16_2:

                db   0EAh       ; JMP far на старый вектор
                                ; прерывания 16h

int16_old       dw   0          ; Область для хранения системного
                dw   0          ; адреса прерывания 16h

demo            endp

code            ends
                end         demonstr
```



# Установка метки тома из программы

*Д. Правиков*

В ряде случаев, как, например, создание набора дистрибутивных дискет прикладного программного комплекса, чтобы не усложнять жизнь разработчикам, бывает необходимо уметь устанавливать метку тома из программы. Поскольку однажды созданная метка тома не может быть уничтожена стандартными средствами (исключая форматирование), а может быть только изменена, то в подобной программе необходимо предусмотреть соответственно две ветви: создания и коррекции.

Если создание метки тома можно реализовать как открытие обычного файла со специальным атрибутом (08h), то для смены метки тома существует специальный алгоритм, описание которого из всех доступных автору источников приведено только в [1]. К сожалению, при изложении алгоритма была допущена ошибка, обнаружение и исправление которой явилось поводом для написания данной заметки. Далее приведен правильный алгоритм, который иллюстрирует программа (листинг). Прежде чем начать его изложение, напомним понятие расширенного FCB, которое в данном случае является ключевым. Расширенный FCB имеет следующую структуру (рис. 1).

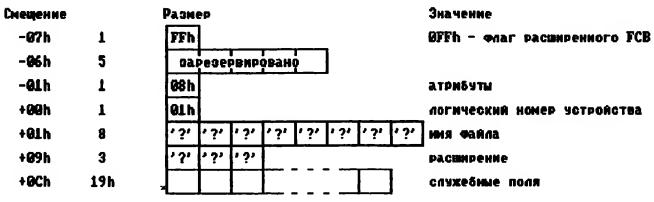



**Рис. 1. Структура блока управления файлами**

Формат служебных полей не приводится, так как он несущественен для данной задачи. На рисунке также приведены те значения полей, задание которых необходимо для поиска метки тома.

Алгоритм смены метки тома следующий:

1. Установить адрес передачи данных (DTA) на 64-байтовый буфер, который DOS использует в качестве рабочей области для функций поиска в директории.

2. Используя расширенный FCB с указанными значениями, выполнить функцию DOS 11h (поиск до первого совпадения). Если регистр AL вернул 0FFh, диск не имеет метки.

3. Если функция 11h вернула в AL ноль, то буфер содержит расширенный FCB с именем тома, расположенным в байтах с 08h до 12h. Заслать новое значение метки по смещению 18h (а не 19h) относительно начала буфера.

4. Поместив адрес буфера в регистры DS:DX, вызвать функцию 17h (переименование файла). Если функция вернула ноль, то изменение метки тома прошло успешно. Если AL содержит 0FFh, то либо Ваша программа, либо операционная система содержит ошибку.

5. Если диск не имеет метки, заменить '?' на 11-символьное имя тома и вызвать функцию 16h (создать файл), передав в DS:DX адрес расширенного FCB.

Данная процедура всегда помещает метку тома в корневую директорию, независимо от текущей директории на диске. ■

**Литература**

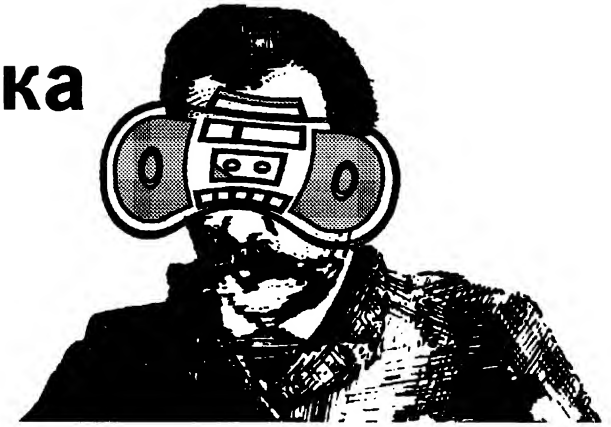
**1. Ray Duncan. Advanced MS-DOS.– Microsoft Press, 1986.**

**Листинг**

```
; Программа изменения метки тома гибкого диска
;
                .MODEL  SMALL
                .STACK  100h
                .DATA
;---------- Шаблон расширенного FCB для поиска метки тома --
xFCB            db      0FFh
                db      5 dup (0)
                db      08h
                db      1                       ; 1 = 'A:'
                db      11 dup ('?')
                db      25 dup (?)
;---------- Область передачи данных при дисковых операциях
buffer          db      64 dup (?)
;---------- Новая метка тома ----------
dlabel          db      'MYLABEL    '
                .CODE
Start:
                mov     ax,@data
                mov     ds,ax
;---------- Переустановка DTA ----------
                mov     dx,OFFSET buffer
                mov     ah,1Ah
                int     21h
;---------- Поиск метки тома ----------
                mov     dx,OFFSET xFCB
                mov     ah,11h
                int     21h
                cmp     al,0FFh
                je      no_label
;---------- Поместить новую метку в буфер ----------
                push    ds
                pop     es
                mov     si,OFFSET dlabel
                mov     di,OFFSET buffer
                add     di,18h                  ; Важная константа!
                mov     cx,11
                rep     movsb
;---------- Изменение метки тома ----------
                mov     dx,OFFSET buffer
                mov     ah,17h
                int     21h
no_label:
;---------- Выход из программы ----------
                mov     ax,4C00h
                int     21h
                END     Start
```

■

# Стереографика



*Д. Солдатенков*

Теперь уже трудно установить, кому первому пришла в голову мысль использовать синий и красный светофильтры для получения стереоэффекта при рассматривании красно-синего изображения. В детстве у меня была книга Анны Владимировны Дуровой «Театр зверей», в которую вкладывались красно-синие очки, изготовленные из целлулоида и картона. Иллюстрации в книге были тоже красно-синие. Инструкция по использованию книги гласила: «Все цветные картинки в книжке надо рассматривать через очки, которые лежат в кармашке на последней странице.

Положите книжку перед собой и слегка приподнимите ее верхний край. Наденьте очки так, чтобы правым глазом смотреть через синее стеклышко, а левым через красное, и Вы заметите, как постепенно оживает картинка. Вы ясно увидите, что находится впереди и что — позади, картинки оживут и сойдут со страниц книжки.

Важно найти правильное расстояние между картинкой и глазами — оно для каждого из вас различно.

Рассматривать картинки надо при дневном свете или при ярком и ровном электрическом освещении, но не на солнце.

Если вы носите очки, то цветные очки можно надеть поверх обычных или поместить их между глазами и обычными очками.

В перевернутом виде или сбоку рассматривать картинки нельзя...».

Впоследствии при экспериментах с компьютерной стереографикой выяснилось, что каждый пункт этой инструкции связан с одной из причин, по которой возможна потеря стереоэффекта.

Книга «Театр зверей» была выпущена издательством «Детский мир» в 1963 г. и стоила 55 копеек.

Два года назад мой коллега Парфентьев А.Н. пожаловался, что после того, как он прочитал статью об использовании англичанами разноцветных очков для вывода на экран информации с компьютерного томографа, ему очень хочется увидеть стереоэффект на экране компьютера.

Красное стекло было взято от фотоувеличителя, с синим было труднее. Я взял кусочек оргстекла и несколько раз покрыл его синей тушью. Наспех слепил программку, которая выводила на экран стереоизображение пирамиды. Даже через такие несовершенные очки стереоэффект наблюдался очень хорошо. Синее оргстекло заменили на светофильтр от набора для цветной фотопечати. Теперь объем воспринимался без какого-либо напряжения.

Тестовая программа с вращающейся пирамидой, полный исходный текст которой приводится в статье, демонстрировалась на разных экранах разным людям. 80% наблюдателей хорошо и сразу воспринимали стереоэффект. 10% воспринимали эффект не сразу, но, один раз увидев его, уже не теряли. Я думаю, это связано с тем, что человек привык смотреть на поверхность экрана, а для того чтобы почувствовать стереоэффект, надо сфокусировать взгляд на изображении, которое находится в глубине. 10% не могли увидеть стереоэффект совсем.

Часто осложняли дело собственные очки наблюдателя, разность в плотности светофильтров, плохая настройка монитора, неправильное положение наблюдателя у экрана. Я сам воспринимаю стереоэффект настолько ярко, что мне кажется странным, как это некоторые люди его не видят.

Была создана игра, в которой использовался стереоэффект. В игре требуется, управляя вертолетом, сбрасывать бомбочки на танки. Без очков играть гораздо труднее. Очень забавно наблюдать, как в пылу сражения игрок старается заглянуть за край экрана.

Используя 256 цветов, можно поддерживать для стереоизображений 16 градаций яркости. Та-

кие изображения выглядят довольно реалистично. Пример был опубликован на цветной вкладке журнала «Наука и жизнь» (№ 7, 1992) вместе со статьей «Что там в заэкранье?».

Оказалось, что работы над стереографикой ведутся многими командами в Москве и за ее пределами. На выставке «Графикон» были три организации, представлявшие стереографику, причем одна из них уже вела мелкосерийный выпуск стереоочков. Когда представитель одной израильской фирмы поместил в газете объявление о том, что желает распространять советские компьютерные игры, ему принесли сразу две игры со стереоэффектом.

Несмотря на то, что программы, использующие стереоэффект, вызывают большой интерес, коммерческим успехом они не пользуются. Может быть, это связано с потерей цветов, а может быть с тем, что в красно-синих очках глаза очень быстро устают. Тем не менее, возможно, существуют приложения, где стереоэффект будет очень полезен.

Технология создания стереоизображений довольно проста. Надо рассчитать обычное трехмерное изображение для каждого глаза и потом нарисовать одно из них красным цветом, а другое — синим (рис. 1).

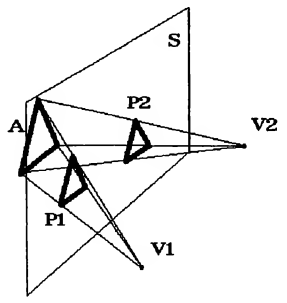



**Рис. 1**

**S — плоскость экрана,**
**A — трехмерный объект,**
**V1, V2 — положения глаз,**
**P1, P2 — проекции для левого и правого глаза.**

О том, как построить проекцию трехмерного объекта на плоскость, подробно рассказывалось в статье «Трехмерная графика» в 1 номере журнала «Монитор» за 1993 г. Если внести в исходные тексты, приводимые в этой статье, небольшие изменения, то получится программа для вывода стереоизображений (листинг 1).

Программа написана на языке Turbo Pascal и использует его стандартную графику с одним небольшим дополнением: при выводе линии одним цветом линия, нарисованная другим цветом, не стирается. ■

ELITSoft

**Пользователям вычислительной техники Акционерное общество "ЭЛИТ СОФТ" предлагает**

Агрегаты Бесперебойного Питания

(Uninterruptable Power Source)

*Основные характеристики*

- Мощность 900 ВА
- Время автономной работы 40 мин.
- Время переключения 5-7 мсек.
- Рабочий диапазон 180-240 В
- Гарантия 1 год
- Послегарантийное обслуживание

*В конструкции предусмотрены*

- Фильтрация сетевых помех
- Звуковая и световая сигнализация
- Индикация перегрузки
- Индикация уровня заряда батарей
- Герметичные, необслуживаемые аккумуляторные батареи

*Поставка и оплата*

Поставка со складов в Москве, Санкт-Петербурге, Чите, Рязани. Любая удобная Вам форма оплаты.

**Адрес: 105215 Москва, 11-я Парковая, 44, корп. 1**

**Телефон:**
**секретариат: (095) 965-35-29, 532-96-14**
**отдел сбыта: (095) 965-35-28, 965-35-30**
**Факс (095) 965-35-28, 965-35-30**
**Телетайп 205211 КТУР**

*Мы продаем компьютеры!*

**Листинг**

```
uses Graph, Crt;

type
    TLoc = record
        X, Y, Z: Real;
    end;
    TLocArr = array [1..1024] of TLoc;
    PLocArr = ^TLocArr;

    TPln = record
        p: array [1..3] of Word;
        Dist: Real;
    end;
    TPlnArr = array [1..1024] of TPln;
    PPlnArr = ^TPlnArr;

    TSc = record
        X, Y: Integer;
    end;
    TScArr = array [1..1024] of TSc;
    PScArr = ^TScArr;

const
    GrToRad: Real = 3.1416/180;          { Перевод градусов в радианы }
    GD: Integer = 3;                     { Графический режим }
    GM: Integer = 1;
    VPage0: Pointer = Ptr ($A000, 0);    { Указатели на видеостраницы }
    VPage1: Pointer = Ptr ($A800, 0);
    RLook: TLoc = (X : 0; Y : 25; Z :-550);
    LLook: TLoc = (X : 0; Y : -25; Z :-550);

    PirIniPoi: array [1..5] of TLoc = ({ Точки пирамиды }
    (X : 0; Y : 50; Z :-50),
    (X : 0; Y : 50; Z : 50),
    (X : 0; Y :-50; Z : 50),
    (X : 0; Y :-50; Z :-50),
    (X : 90; Y : 0; Z : 0)
    );

    PirPln: array [1..4] of TPln = ({ Грани пирамиды }
    (p : (1,2,5); Dist : 0),
    (p : (2,3,5); Dist : 0),
    (p : (3,4,5); Dist : 0),
    (p : (4,1,5); Dist : 0)
    );

var
    Look: TLoc;              { Положение наблюдателя }
    PirPoi: TLocArr;         { Развернутый массив точек }
    PirSc: TScArr;           { Массив точек на экране }

function Xplan ( X, Z: Real): Integer;
begin
    Xplan := Trunc (Look.Z*X / (Look.Z-Z));
end;

function Yplan ( Y, Z: Real): Integer;
begin
    Yplan := Trunc (Look.Z*Y / (Look.Z-Z) + Look.Y);
end;

procedure OnScreen ( X, Y: Integer; var ScrX, ScrY: Integer);
{ Координаты точек на экране }
begin
    ScrX := Trunc (Y*1.3 + 320);
    ScrY := 175 - X;
end;

procedure FormPoi (Risk, dX, dY, dZ: Integer;
                   IniPoi, Poi: PLocArr; NPoi: Word);
{ Разворачиваем точки }
var
    i: Integer;
    CosA, SinA: Real;
    A: Real;
begin
    A := Risk*GrToRad;
    CosA := Cos (A);
    SinA := Sin (A);
    for i := 1 to NPoi do
        with IniPoi^[i] do begin
            Poi^[i].X := X + dX;
            Poi^[i].Z := Z * CosA - Y * SinA + dZ;
            Poi^[i].Y := Z * SinA + Y * CosA + dY - Look.Y;;
        end;
end;

procedure FormScPoi (Poi: PLocArr; ScPoi: PScArr; NPnt: Word);
{ Считываем положение точек на экране }
var
    i: Integer;
begin
    for i := 1 to NPnt do
        with Poi^[i] do
            OnScreen ( Xplan (X, Z), Yplan (Y ,Z), ScPoi^[i].X, ScPoi^[i].Y);
end;

procedure DrawPlan (ScPoi: PScArr; Pln: PPlnArr; NPln: Word; Color:
Byte);
{ Рисуем грани }
var
    i, j: Integer;
begin
    or i := 1 to NPln do begin
        for j := 1 to 2 do begin
            Port [$3C4] := 2;
            Port [$3C5] := Color;
            Line (ScPoi^[Pln^[i].p[j]].X, ScPoi^[Pln^[i].p[j]].Y,
                ScPoi^[Pln^[i].p[j+1]].X, ScPoi^[Pln^[i].p[j+1]].Y);
        end;
        Port [$3C4] := 2;
        Port [$3C5] := Color;
        Line (ScPoi^[Pln^[i].p[3]].X, ScPoi^[Pln^[i].p[3]].Y,
                    ScPoi^[Pln^[i].p[1]].X, ScPoi^[Pln^[i].p[1]].Y);
    end;
end;

procedure Show;
{ Показываем невидимую страницу }
begin
    Port [$3CE] := 5;
    Port [$3CF] := 1;
    Move (VPage1^, VPage0^, 80*350);
    Port [$3CE] := 5;
    Port [$3CF] := 0;
end;

var
    Ch: Char;
    i, j: Integer;

begin
    InitGraph (GD, GM, '');
    SetActivePage (1);
    j := 150;
    repeat
    for i := 0 to 35 do
        begin
            j := i * 10;
            SetFillStyle (1, 0);
            Bar (0, 0, 639, 349);
            { Для правого глаза }
            Look := RLook;
            FormPoi (j, -60, 0, -250+(700-j*2), @PirIniPoi, @PirPoi, 5);
            FormScPoi (@PirPoi, @PirSc, 5);
            SetColor (4);
            DrawPlan (@PirSc, @PirPln, 4, 4);
            { Для левого глаза }
            Look := LLook;
            FormPoi (j, -60, 0, -250+(700-j*2), @PirIniPoi, @PirPoi, 5);
            FormScPoi (@PirPoi, @PirSc, 5);
            SetColor (1);
            DrawPlan (@PirSc, @PirPln, 4, 1);

            Show;
        end;
        Delay (1000);
    until KeyPressed;
    Ch := ReadKey;
    CloseGraph;
end.
```

# Библиотека графических курсоров «мыши»

*А.Аброськин*

Информация

**Журнал МОНИТОР**

**создает дилерскую сеть...**

Своим региональным представителям «Монитор» предлагает:

- Оптовые партии нашего журнала для распространения.
- Программные продукты для реализации в своем регионе с широкой системой скидок.
- Регулярную публикацию рекламных объявлений на льготных условиях.
- Оперативную информацию о состоянии рынка.

Для получения более подробной информации вышлите по адресу: 117419 Москва, а/я 765 письмо, содержащее краткие сведения о Вашей фирме, регионе, в котором Вы работаете, о том, какие программные продукты Вас интересуют, какое количество экземпляров журнала Вы хотели бы реализовывать.

Справки по тел. (095) 238-99-33.
Факс: (095) 238-32-78.

При разработке программ, активно использующих в своей работе позиционный манипулятор «мышь», нередка ситуация, когда, по замыслам разработчика программы, форма графического курсора «мыши» должна соответствовать какой-то определенной операции или отражать текущее состояние системы, как, например, это делает Microsoft WINDOWS.

Отсутствие библиотек курсоров «мыши» заставляет программиста вручную заниматься такой рутинной работой, как кодирование битовых масок курсора и экрана.

Хотя «творческий» процесс кодирования — обычное дело профессионалов, но начинающего программиста, особенно если он занимается разработкой своих программ на языке высокого уровня, могут свести с ума эти бесконечные ряды ноликов и единичек. Как следствие, результат оказывается далеко не прямо пропорциональным затраченному времени.

Журнал «Монитор» уже писал о реализации интерфейса «мыши» из программ на Turbo Pascal [1]. К сожалению, процедура SetGraphMouseShape, представленная в библиотеке процедур и функций для обслуживания драйвера «мыши», позволяет иметь только два типа графических курсоров. Для большинства приложений этого вполне достаточно. Однако если по каким-либо причинам Вам захочется иметь что-нибудь еще, то придется немного попотеть и самому внести необходимые изменения в процедурную часть программы.

В предлагаемом Вашему вниманию листинге процедуры SetGraph MouseShape подобные изменения уже произведены. В частности, предусмотрен выбор любого из 11 типов курсоров «мыши». При передаче неверного параметра процедура автоматически будет использовать тип графического курсора Shape_1. Значения битовых масок даются в шестнадцатиричном формате. Это, конечно же, не лучшим образом сказывается на наглядности и читаемости программы, но в то же время позволяет избежать досадных ошибок и опечаток при вводе этих значений вручную. Описание и внешний вид курсоров даются в комментариях листинга, что, впрочем, не мешает Вам, не откладывая до завтра, прямо сегодня набрать эти цифры и посмотреть на курсоры сквозь призму Вашего монитора.

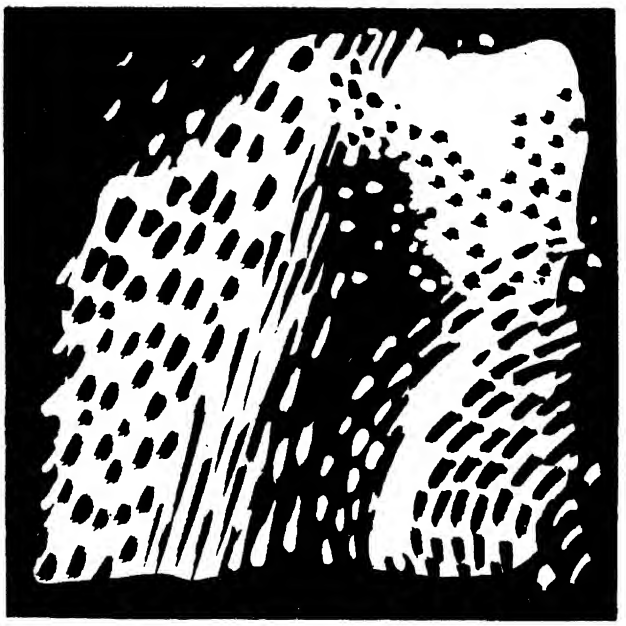
*Люди! Будьте бдительны!*
*Если Вы уже установили Borland Pascal 7.0, замените*
*Proc SetGraphMouseShape*
*на*
*SetGraphMouseShape Proc.* ■

**Литература**

**1. В.Водолазкий, В.Головин, Е.Короткова. Паскаль+Ассемблер=... или как использовать «мышь» в графических режимах// Монитор.– 1992.– № 5.**

**Листинг 1**

```
        NAME    CURSOR

        MODEL   TPASCAL

        .RADIX  16

        LOCALS

        .CODE

        PUBLIC  SETGRAPHMOUSESHAPE

PROC    SETGRAPHMOUSESHAPE  SHAPE:WORD

        PUSH    DS          ;Сохраняем сегмент данных,
        PUSH    CS          ;настраиваем сегментные регистры
        PUSH    CS          ;<ES> и <DS> на сегмент кода
        POP     DS
        POP     ES

        ASSUME DS: CODE

        MOV     SI,[SHAPE]  ;Загружаем номер графического образа

        CMP     SI,12D      ;Максимальный размер?
        JB      @@1
        MOV     SI,1        ;Коррекция недопустимого номера
@@1:
        DEC     SI          ;Формируем смещение адреса курсора
        SHL     SI,1        ;в таблице

        MOV     DX,CS:SHAPE_ADR[SI] ;Загружаем адрес курсора в DX
        MOV     BX,CS:HOT_X[SI]  ;Загружаем в регистры <BX> и <CX>
        MOV     CX,CS:HOT_Y[SI]  ;координаты "горячего пятна"

        MOV     AH,09       ;Вызов функции "Установка типа курсора"
        INT     33          ;для графического режима

        POP     DS          ;Восстанавливаем адрес сегмента данных
        RET                 ;Выход из процедуры

;-------------------------------------------------------------------
;
; HOT_X, HOT_Y - координаты "горячего пятна" графических курсоров
;
;       №       1   2   3   4   5   6   7   8   9   10  11
;-------------------------------------------------------------------
HOT_X   DW      2D, 15D,13D,8D, 8D, 8D, 8D, 8D, 7D, 8D, 8D
HOT_Y   DW      1D, 1D, 3D, 7D, 7D, 5D, 5D, 5D, 7D, 8D, 8D

;-------------------------------------------------------------------
;
; SHAPE_ADR - адреса масок экрана и курсоров
;
;-------------------------------------------------------------------
SHAPE_ADR  DW SHAPE_1
           DW SHAPE_2
           DW SHAPE_3
           DW SHAPE_4
           DW SHAPE_5
           DW SHAPE_6
           DW SHAPE_7
           DW SHAPE_8
           DW SHAPE_9
           DW SHAPE_10
           DW SHAPE_11

;-------------------------------------------------------------------
; Курсор вида  "перст указующий" -
;              указательный палец правой руки (кулака),
;              направленный вертикально вверх
;-------------------------------------------------------------------
SHAPE_1:
DW 08FFFH,027FFH,077FFH,077FFH,077FFH,077FFH,07FFFH,07124H
DW 07492H,076DAH,076DAH,076D6H,07FFEH,03FFCH,09FF9H,0C003H
DW 00000H,02000H,07000H,07000H,07000H,07000H,07000H,07000H
DW 07492H,076DAH,076DAH,076D6H,07FFEH,03FFCH,01FF8H,00000H

;-------------------------------------------------------------------
; Курсор вида  "перст указующий" (2) -
;              то же самое, но для левой руки, причем изображение внутри
;              контура руки инвертировано по отношению к фону
;-------------------------------------------------------------------
SHAPE_2:
DW 0FFF1H,0FFE0H,0FFE0H,0FFE0H,0FFE0H,0FFE0H,02480H,00000H
DW 00000H,00000H,00000H,00000H,00000H,08000H,0C001H,0C003H
DW 00000H,00004H,0000EH,0000EH,0000EH,0000EH,0000EH,0492EH
DW 05B6EH,05B6EH,04B6EH,0692EH,03FFEH,01FFCH,01FF8H,00000H

;-------------------------------------------------------------------
; Курсор вида  "ладонь" -
;              изображение берущей человеческой ладони
;              (что-то от PaintBrush)
;-------------------------------------------------------------------
SHAPE_3:
DW 0FE67H,0FC47H,0FCCCH,0C888H,0C999H,09911H,09333H,03223H
DW 02007H,00007H,0001EH,0003CH,08038H,08001H,0C003H,0C01FH
DW 00198H,003B8H,00333H,03777H,03666H,066EEH,06CCCH,0CDDCH
DW 0DFF8H,0FFF8H,0FFE1H,0FFC3H,07FC7H,07FFEH,03FFEH,03FE0H

;-------------------------------------------------------------------
; Курсор вида  "песочные часы"
;              в стиле WINDOWS
;-------------------------------------------------------------------
SHAPE_4:
DW 00000H,00000H,08001H,0C003H,0E007H,0F00FH,0F81FH,0F81FH
DW 0F81FH,0F81FH,0F00FH,0E007H,0C003H,08001H,00000H,00000H
DW 00000H,07FFEH,00000H,01FF8H,00FF0H,003C0H,00000H,003C0H
DW 00000H,003C0H,007E0H,00FF0H,01FF8H,00000H,07FFEH,00000H

;-------------------------------------------------------------------
; Курсор вида  "песочные часы" (2) -
;              вариант предыдущего с закрашенной внутренней полостью
;              колбы
;-------------------------------------------------------------------
SHAPE_5:
DW 00000H,07FFEH,08001H,0DFFBH,0EFF7H,0F3CFH,0F81FH,0FBDFH
DW 0F81FH,0FBDFH,0F7EFH,0EFF7H,0DFFBH,08001H,07FFEH,00000H
DW 00000H,07FFEH,00000H,01FF8H,00FF0H,003C0H,00000H,003C0H
DW 00000H,003C0H,007E0H,00FF0H,01FF8H,00000H,07FFEH,00000H

;-------------------------------------------------------------------
; Курсор вида  "восклицательный знак"
;              очень похож на WOW!
;-------------------------------------------------------------------
SHAPE_6:
DW 0F3CFH,0F81FH,0F00FH,0F00FH,0F00FH,0F00FH,0F00FH,0F00FH
DW 0F81FH,0F81FH,0FC3FH,0FFFFH,0FC3FH,0FC3FH,0FC3FH,0FC3FH
DW 00000H,00180H,003C0H,007E0H,007E0H,007E0H,007E0H,003C0H
DW 003C0H,00180H,00000H,00000H,00000H,00180H,00180H,00000H

;-------------------------------------------------------------------
; Курсор вида  "вопросительный знак"
;-------------------------------------------------------------------
SHAPE_7:
DW 0FC0FH,0F007H,03003H,0E003H,0C083H,0C103H,0C307H,0C20FH
DW 0FC1FH,0FC1FH,0FC3FH,0FC3FH,0FFFFH,0FC3FH,0FC3FH,0FC3FH
DW 00000H,001E0H,007F0H,00E38H,00C18H,01830H,01860H,00040H
DW 000C0H,00180H,00000H,00000H,00180H,00180H,00180H,00000H

;-------------------------------------------------------------------
; Курсор вида  "вопросительный знак"
;              с инвертированием
;-------------------------------------------------------------------
SHAPE_8:
DW 0FC0FH,0F1E7H,0E7F3H,0EE3BH,0CC9BH,0D933H,0DB67H,0C24FH
DW 0FCDFH,0FD9FH,0FDBFH,0FC3FH,0FFFFH,0FDBFH,0FDBFH,0FC3FH
DW 00000H,001E0H,007F0H,00E38H,00C18H,01830H,01860H,00040H
DW 000C0H,00180H,00180H,00000H,00000H,00180H,00180H,00000H

;-------------------------------------------------------------------
; Курсор вида  "диагональное перекрестие":
;              в центре - маленькая дырочка
;-------------------------------------------------------------------
SHAPE_9:
DW 00FF0h,007E0h,003C0h,08181h,0C003h,0E007h,0F00Fh,0F99Fh
DW 0F00Fh,0E007h,0C003h,08181h,003C0h,007E0h,00FF0h,01FF8h
DW 00000h,06006h,0300Ch,01818h,00C30h,00660h,00240h,00000h
DW 00240h,00660h,00C30h,01818h,0300Ch,06006h,04002h,00000h

;-------------------------------------------------------------------
; Курсор вида  "прямоугольное перекрестие":
;              внутри перекрестия виден фон
;-------------------------------------------------------------------
SHAPE_10:
DW 0FC1Fh,0FC1Fh,0FC1Fh,0FC1Fh,0FC1Fh,0FC1Fh,08080h,08080h
DW 083E0h,08080h,08080h,0FC1Fh,0FC1Fh,0FC1Fh,0FC1Fh,0FC1Fh
DW 00000h,00140h,00140h,00140h,00140h,00140h,00140h,03F7Eh
DW 00000h,03F7Eh,00140h,00140h,00140h,00140h,00140h,00000h

;-------------------------------------------------------------------
; Курсор вида  "прямоугольное перекрестие" -
;              то же самое, но только чуть пошире,
;              побольше и с инвертированной дырочкой посередине
;-------------------------------------------------------------------
SHAPE_11:
DW 0FFFFh,0F88Fh,0F88Fh,0F88Fh,0F88Fh,08080h,08080h,08080h
DW 0FFFFh,08080h,08080h,08080h,0F88Fh,0F88Fh,0F88Fh,0F88Fh
DW 00000h,00220h,00220h,00220h,00220h,00220h,07EBFh,00080h
DW 003E0h,00080h,07EBFh,00220h,00220h,00220h,00220h,00220h

ENDP

END
```



# Выбери цвета...

*И. Завалей,*
*И. Михайлов*

Часто требуется снабдить разрабатываемую программу механизмом, с помощью которого пользователь может подобрать цветовую гамму пользовательского интерфейса в соответствии со своими индивидуальными представлениями о прекрасном. Один из возможных путей реализации такого механизма, созданный для работы с EGA адаптером, описан ниже. В основе лежит идея управления уровнями интенсивности основных цветов RGB палитры [1].

Известно, что каждый из трех основных цветов RGB палитры (красный, синий и зеленый) может иметь один из четырех возможных уровней интенсивности: 0, 1/3 max, 2/3 max и max. Каждый из 64 возможных цветов EGA адаптера получается комбинацией основных цветов, имеющих разный уровень интенсивности.

Алгоритм настройки цветов можно представить в виде последовательности действий.

1. Выбрать цвет, который желательно изменить.
2. Определить номер регистра текущей палитры, соответствующий выбранному цвету.
3. Определить номер цвета RGB палитры, соответствующий этому регистру.
4. Определить уровни интенсивности основных цветов.
5. Изменить уровни интенсивности желательным образом.
6. Установить в регистр текущей палитры измененный цвет.

Наиболее просто цветовую настройку программы осуществлять, когда на экране находится некоторое обобщенное представление пользовательского интерфейса, а пользователь с помощью «мыши» может выбрать тот цвет, который желает изменить. Описанная схема реализована в процедуре UserColor модуля OwnColor, написанного в среде TURBO PASCAL 5.0 (листинг 1).

### ОПИСАНИЕ МОДУЛЯ

**Глобальный тип**

```
TypePalette=array[0..15] of byte;
```

**Глобальная константа**

```
StandartPalette:548l 4l
StandartPaletteTypePalette=
        (0, 1, 2, 3, 4, 5, 20, 7, 56, 57,
        58, 59, 60, 61, 62, 63);
```

**Глобальная переменная**

```
FPal:file of TypePalette;
```

### ПРОЦЕДУРЫ И ФУНКЦИИ МОДУЛЯ

**1. Procedure**

```
UserColor (var palette:TypeOwnPalette;
           pr:byte);
```

возвращает переменную palette после установки цветов.

Признак

```
pr=0 — все нормально,
pr=1 — нет драйвера «мыши».
```

**2. Procedure**

```
RestoreStandartPalette;
```

восстанавливает стандартную палитру.

**3. Function**

```
SaveColor (FileName:string;
           palette:TypeOwnPalette):boolean;
```

функция сохранения палитры пользователя в файле.
Входные переменные:

FileName — имя файла, составленное по правилам DOS;
palette — сохраняемая палитра.

Возвращает True в случае успешной записи и False в случае ошибочного задания FileName.

**4. Function**

```
LoadColor (FileName:string;
           var palette:TypeOwnPalette):
           boolean;
```

функция чтения палитры из файла.
Входная переменная

FileName — имя файла, составленное по правилам DOS.

Выходная переменная

palette — прочитанная из файла палитра.

Возвращает True в случае успешного прочтения и False, когда файл FileName не найден.

**5. Procedure**

```
SetOwnPalette(palette:TypeOwnPalette);
```

процедура установки прочитанной палитры.
Входная переменная

palette — палитра, которую надо установить.

Модуль предназначен для работы как в графическом, так и в текстовом режимах адаптера EGA. Указателем «мыши» выбирается на экране поле, цвет которого надо изменить, и нажимается левая кнопка.

Изменение цвета осуществляется клавишами управления курсором, причем уровнем красного цвета управляет клавиша «влево», уровнем синего — клавиша «вниз», уровнем зеленого — клавиша «вправо». В текстовом режиме указанные действия приведут к изменению цвета фона. Для изменения цвета текста используются те же клавиши вместе с клавишей Shift. При желании процедуру легко изменить и использовать для управления цветом три кнопки «мыши» или любой другой механизм. Для удобства управления подбором цветов значения уровней интенсивности можно вывести на экран.

В листинге 2 приведен пример простой программы, использующей модуль OwnColor. ■


**Литература**

1. А.Тихонов. Показать картинку// Монитор.– 1992.– № 3.



INTERLINK

Официальный диллер SIREX, USA
фирма ИНТЕРЛИНК предлагает

386dx
486sx
486dx
486dx2

MHZ 25/33/50/66
BUS ISA/LocalBUS/EISA
RAM 4/8/16/32/64/128 Mb
HDD 120/240/340/670/1.2
MONITOR 14"/15"/17"/20"
NETWORK - NOVELL
SCANER A4/A3/A0/HAND
PRINTER LASER/JET/DOT
PLOTER A4/A3/A2/A1/A0
DIGITIZER A4/A2/A1/A0

*Опыт эксплуатации компьютеров SIREX в России доказал незаменимость их при использовании на особо важных работах. Наши специалисты помогут подобрать наиболее удобный комплект оборудования под ваши задачи. Поставка компьютера или периферии из США у нас займет всего 15 дней.*

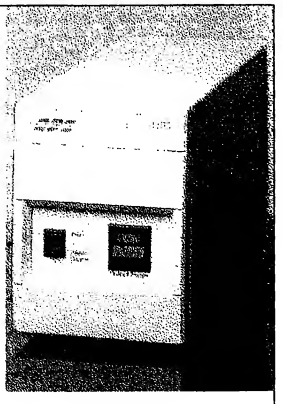
SIREX

*Всегда высшего качества*
СДЕЛАНО С ГОРДОСТЬЮ В АМЕРИКЕ

453-11-18, 456-66-44, 456-83-69

Источники бесперебойного питания

СЛУЖАТ ДЛЯ ЗАЩИТЫ КОМПЬЮТЕРНОЙ СИСТЕМЫ ОТ ПЕРЕПАДОВ НАПРЯЖЕНИЯ И СБОЕВ В ЭЛЕКТРОСЕТИ



КАРАТ-2000
ОФИЦИАЛЬНЫЙ ДИСТРИБЬЮТЕР "AMERICAN POWER CONVERSION"

Телефон: (095) 200-1397, 200-1398
Телефакс: (095) 200-1393
Адрес: Москва, Садовая-Самотечная ул., 5

Региональные дилеры:

| | | |
|---|---|---|
| С.-Петербург | АО "СППКК" | 515-2705 |
| | ТОО "НТЦ МИТ" | 298-0000 |
| Барнаул | НПП "ВАРИАНТ" | 26-04-51 |
| Екатеринбург | АО "УРАЛКОМ" | 44-09-93 |
| Иркутск | ВП "ГРАДИЕНТ" | 23-30-92 |
| Калининград | НПК "СИМВОЛ" | 21-52-74 |
| Новосибирск | ТОО "МАЙКРОЛЭБ" | 35-71-34 |
| Сыктывкар | НПИКК "ГАРАНТ" | 2-56-26 |
| Тула | МНТП "МИРТЭКС" | 31-78-72 |
| Тюмень | АО "РЕГИОН" | 25-05-25 |
| Ульяновск | ВЭ ПКП "АГРОТЭК" | 34-34-26 |
| Уфа | НЦ "ВЕКТОР" | 52-91-32 |

**Листинг 1**

```
Unit OwnColor;
INTERFACE
uses dos,graph,crt;
type TypeOwnPalette = array[0..15] of byte;
const StandartPalette:TypeOwnPalette =
      (0,1,2,3,4,5,20,7,56,57,58,59,60,61,62,63);
var
      FPal:file of TypeOwnPalette;
      rr:registers;
Procedure UserColor(var palette:TypeOwnPalette; var priz:byte);
      { Процедура настройки палитры пользователя.
      Настройка производится нажатием клавиш управления
      курсором (для изменения цвета в графическом режиме
      и изменения цвета фона в текстовом режиме)
      или одновременным нажатием клавиши Shift и клавиш
      управления курсором для изменения цвета текста в
      текстовом режиме. Цвет для настройки определяется
      положением указателя "мыши" в момент нажатия левой
      кнопки. Фиксация выбранного цвета производится
      нажатием клавиши Enter. Выход из процедуры настройки
      происходит нажатием правой кнопки "мыши" или клавиши
      Esc }

Function SaveColor(FileName:string;
            palette:TypeOwnPalette):boolean;
{ Функция сохранения палитры пользователя в файле }
Function LoadColor(FileName:string;
                  var palette:TypeOwnPalette):boolean;
{ Функция чтения палитры из файла }

Procedure SetOwnPalette(palette:TypeOwnPalette);
{ Процедура установки прочитанной палитры }

Procedure RestoreStandartPalette;
{ Процедура восстановления стандартной палитры }

IMPLEMENTATION

Procedure RestoreStandartPalette;
begin
      SetOwnPalette(standartpalette);
end;

Function FileExist(Filename : string) : boolean;
{ Внутренняя процедура: поиск файла }
var flen : byte absolute FileName;
begin
      if Length(FileName) = 0 then fileexist : = false
      else
      with rr do
      begin
            inc(flen); FileName[flen]: = #0;
            ah: = $43; al: = 0; ds: = seg(filename);
            dx: = ofs(filename[1]); intr($21,rr);
            fileexist: = (not (flags mod 2 = 1)) and (ioresult = 0);
      end
end;

Procedure SetOwnPalette(palette:TypeOwnPalette);
var i:integer;
begin
      with rr do
      for i: = 0 to 15 do
      begin
            ah: = $10; al: = 0;
            bl: = i; bh: = palette[i];
            intr($10,rr);
      end;
end;

Function SaveColor(FileName:string;
            palette:TypeOwnPalette):boolean;
begin
      savecolor: = true;
      assign(fpal,filename); rewrite(fpal);
      write(fpal,palette); close(fpal);
      if ioresult < > 0 then savecolor: = false;
end;

Function LoadColor(FileName:string;
            var palette:TypeOwnPalette):boolean;
begin
      loadcolor: = true;
      if fileexist(FileName) then
      begin
            assign(fpal,FileName); reset(fpal);
            read(fpal,palette);
            close(fpal);
      end
      else loadcolor: = false;
end;
Procedure usercolor(var palette:TypeOwnPalette;
            var priz:byte);
{ Константы для определения уровней интенсивности }
{ основных цветов RGB палитры }
const
      r:array[1..4] of byte = (0,32,4,36);
      g:array[1..4] of byte = (0,16,2,18);
      b:array[1..4] of byte = (0,8,1,9);
var
      i,x,y:integer;
      r1t,g1t,b1t,r1,g1,b1:byte;
      but:byte;
      col,ct:byte;
      pr:byte;
      ch:char;
      cc1,cc:shortint;
      bool:boolean;
      mode:(graph,text);
{ переменная для анализа состояния нажатых клавиш Shift }
      kla:word absolute $0:$0417;
      ekr:array[1..25,1..80] of word absolute $B800:$0000;
{ видеобуфер }
begin
      { Инициализация драйвера "мыши" }
      rr.ax: = 0; intr($33,rr);
      priz: = 0;
      if rr.ax < > $FFFF then
      begin
            priz: = 1; exit; { Нет драйвера "мыши" }
      end;
      rr.ax: = 1; intr($33,rr); { Указатель курсора виден }
      rr.ah: = $f; intr($10,rr);
      if rr.al in [0..3,7] then { Определение видеорежима }
            mode: = text
      else
            mode: = graph;
      repeat
            repeat
                  { Запрос о положении указателя "мыши" }
                  { и статусе кнопок }
                  rr.ax: = 3; intr($33,rr);
                  x: = rr.cx; y: = rr.dx;
                  delay(100);
                  but: = 1;
                  if rr.bx = 1 then but: = 0;
                  { Вход в установщик цветов по }
                  { нажатию левой кнопки "мыши" }
                  if rr.bx = 2 then exit;
                  { Выход из процедуры по }
                  { нажатию правой кнопки "мыши" }
            until but = 0;
            if mode = graph then col: = getpixel(x,y)
            else
            begin
                  { Получение координат }
                  { и цветов фона и текста }
                  { в текстовом режиме }
                  x: = x div 8; y: = y div 8;
                  col: = ekr[y,x] shr 12;
                  ct: = (ekr[y,x] and $0f00) shr 8;
            end;

            { Получение номера изменяемого }
            { регистра основной палитры }
            { для точки экрана с координатами x,y }
            if mode = graph then
                  cc: = palette[col] { в графическом режиме }
            else begin
                  cc: = palette[col];
                  cc1: = palette[ct]; { в текстовом режиме }
            end;
            { Определение уровней интенсивности основных }
            { цветов RGB палитры для изменяемого регистра }
            { основной палитры }
            g1: = 1; r1: = 1; b1: = 1;
```

```
for i:=1 to 4 do
begin
     if cc and 9=b[i] then b1:=i;
     if cc and 18=g[i] then g1:=i;
     if cc and 36=r[i] then r1:=i;
end;
{ То же для цвета текста }
if mode=text then
begin
     g1t:=1; r1t:=1; b1t:=1;
     for i:=1 to 4 do
     begin
          if cc1 and 9=b[i] then b1t:=i;
          if cc1 and 18=g[i] then g1t:=i;
          if cc1 and 36=r[i] then r1t:=i;
     end;
end;
repeat
     repeat until keypressed;
     case mode of
     graph: pr:=0;
     text:
     begin     { Проверка состояния }
          { клавиш Left- и Right-Shift }
          pr:=kla and 3;
          if pr<>0 then pr:=1;
     end;
     end;{ case of mode }
     ch:=readkey; if ch=#0 then ch:=readkey;
     if ch in [#75,#77,#80,#13,#50,#52,#54,#27]
     then
     begin
          if ch=#27 then exit;
          case pr of
          0: { Изменение цвета для графического
режима или цвета фона для текстового режима }
          case ch of
          #75:
          begin r1:=r1+1; if r1>4
               then r1:=1;
          { изменение красного }
          end;
          #77:
          begin b1:=b1+1; if b1>4
               then b1:=1;
          { изменение синего }
          end;
          #80:
          begin g1:=g1+1; if g1>4
               then g1:=1;
          { изменение зеленого }
          end;
          end;{ case of ch }
          1: { Изменение цвета текста
               для текстового режима }
          if mode=text then
          begin
               case ch of
          { Переопределение "белых" клавиш }
          { управления курсором, нажатых }
          { вместе с Shift }
               #50: ch:=#75;
               #52: ch:=#77;
               #54: ch:=#80;
               end;{ case of ch }
               case ch of
               #75: begin r1t:=r1t+1; if
                    r1t>4 then r1t:=1;
               { изменение красного }
               end;
               #77: begin b1t:=b1t+1; if
                    b1t>4 then b1t:=1;
                    { изменение синего }
               end;
               #80: begin g1t:=g1t+1; if
                    g1t>4 then g1t:=1;
                    { изменение зеленого }
               end;
               end; { case of ch }
               end;
          end; { case of pr }
          if ch=#13 then bool:=true else
          bool:=false;
          { Выход из настройщика }
                    { по клавише Enter }
                    palette[col]:=r[r1]+b[b1]+g[g1];
                    { Определение номера }
                    { регистра RGB палитры }
                    { по уровням интенсивности }
                    { основных цветов }
                    rr.ah:=$10; rr.al:=0; rr.bl:=col;
                    { Установка регистра }
                    { RGB палитры в изменяемый }
                    { регистр основной палитры }
                    rr.bh:=palette[col];
                    intr($10,rr);
                    if mode=text then
                    begin
                         palette[ct]:=r[r1t]+b[b1t]+
                              g[g1];
                         rr.ah:=$10; rr.al:=0; rr.bl:=ct;
                         { Установка регистра RGB палитры
                         в изменяемый регистр основной
                         палитры для цвета текста }
                         rr.bh:=palette[ct];
                         intr($10,rr);
                    end;
               end;
          until bool;
     until false;
end; { процедуры usercolor }
begin
end.
```

**Листинг 2**

```
program demo;
uses dos,graph,crt,owncolor;
var i,j:integer;
          palette1,palette:Typeownpalette;
          pr:byte;
          ch:char;
begin
     for i:=0 to 15 do palette[i]:=standartpalette[i];
     clrscr;
     writeln(  'Для демонстрации работы программы',
               'в графическом режиме');
     writeln(  'нажмите пробел, в текстовом режиме -',
               'любую другую клавишу');
     ch:=readkey;
     if ch=#32 then
     begin
          i:= Detect; InitGraph(i,j,'');
          for i:=1 to 16 do
          begin
               setfillstyle(1,i-1);
               bar(1+38*(i-1),100,35+38*(i-1),200);
               { Демонстрационное цветовое поле для
               графического режима }
          end;
     end else
     begin
          textattr:=112; clrscr;
          window(10,5,70,15); textattr:=49;clrscr;
          for i:=1 to 608 do write('A');
          { Демонстрационное цветовое поле для текстового
          режима }
     end;
     { Настройка цветов }
     usercolor(palette,pr);
     if savecolor('pal.pal',palette) then
     begin
     { Запоминание палитры в файле pal.pal текущей
     директории }
     end;
     { Восстановление стандартной палитры }
     restorestandartpalette;
     ch:=readkey;
     { Чтение палитры из файла pal.pal текущей директории }
     if loadcolor('pal.pal',palette1) then
     { Установка прочитанной палитры в случае успешного
     чтения }
     setownpalette(palette1);
     ch:=readkey;
     { Восстановление стандартной палитры }
     rgstorestandartpalette;
end.
```

# Как зарегистрировать программу в США

*М. Корсаков*

В данной статье подробно рассматриваются процедурные вопросы регистрации компьютерных программ в США.

Разработки программистов-профессионалов из бывшего Советского Союза по-прежнему высоко ценятся за рубежом. В последнее время процесс выхода «наших» на мировой рынок ПО объективно активизировался, равно как и увеличилось число желающих воспользоваться их услугами. Важную роль при этом играют вопросы юридической защиты своих программных продуктов, которым, к сожалению, не всегда уделяется достаточное внимание.

Одним из наиболее перспективных рынков для наших специалистов являются Соединенные Штаты Америки. В ранее публиковавшейся в «Мониторе» статье (№ 2, 1992) мы рассматривали основные положения Закона об авторском праве этой страны, а также затронули вопросы, касающиеся его применения к компьютерным программам. В данной статье хотелось бы специально остановиться на вопросах регистрации программ в соответствии с требованиями законодательства США. Такая регистрация программ, не являясь обязательной, оказывается чрезвычайно полезной прежде всего для доказательства своего преимущественного права при исках о нарушении авторских прав на программу. Если же говорить об отношениях наших программистов с иностранными потребителями их ПО, то такая регистрация, учитывая ситуацию в СНГ и часто встречающееся отношение к нам как к источнику дешевых, а то и бесплатных разработок, просто необходима для цивилизованной торговли и защиты своих интересов в случае их нарушения. При этом, однако, необходимо отметить, что прежде всего программа охраняется национальным законодательством (в России — Законом РФ «О правовой охране программ для электронных вычислительных машин и баз данных»), а защита программы в других странах является действенным вспомогательным (но отнюдь не второстепенным!) средством.

Согласно законодательству США, термин «компьютерная программа» определяется как «...набор переменных или команд, прямо или косвенно использующихся в компьютере для получения заданного результата». Защитой пользуются все элементы программы, подпадающие под определение объектов авторского права. К таковым не принадлежат и, соответственно, не охраняются идеи, логическое построение программы, алгоритмы, методика, концепции.

Процедура регистрации состоит в заполнении специальной Формы ТХ (образец которой приводится в конце статьи) и отправке этой формы, чека на десять долларов и образца программы в отдел регистрации Библиотеки Конгресса по адресу:

*Register of Copyrights*
*Library of Congress*
*Washington, D.C. 20 599.*

В Форме ТХ приводятся следующие данные.

1. Название программы, под которым производится регистрация, и сведения об иных названиях, используемых при описании программы. Если программа является частью совместной разработки, необходимо указать название последней.

2. Автор программы: имя, гражданство или место постоянного проживания, дата рождения и, если автор программы к моменту подачи заявки на регистрацию скончался, дата смерти. Автор может использовать псевдоним, по желанию указав свое настоящее имя, а также зарегистрировать работу как анонимную. В случае если программа была разработана в рамках выполнения служебных обязанностей или по договору подряда, автором считается наниматель или заказчик и в графе «Наименование автора» приводятся их полные наименования. В данной графе указывается и степень участия автора в создании программы, например, «...автор всей программы, соавтор, разработчик части программы» и тому подобное.

3. Дата окончания создания программы и дата и место первой публикации (то есть публичного распространения копий), если таковая имела место к моменту подачи заявки на регистрацию.

4. Сведения о владельце авторских прав на программу (имя и адрес). Им может быть сам автор, его наниматель или подрядчик, а также лица, которым автор передал права на свою разработку (в последнем случае указывается способ приобретения авторских прав на программу, например, покупка).

5. Сведения о предыдущих регистрациях программы, если таковые имели место, и объяснение причин подачи заявки на регистрацию.

6. В случае если программа является компиляцией или творческой переработкой ранее существовавших модулей, указывается перечень использованных материалов, а также характер изменений, внесенных автором.

7. Седьмая графа Формы ТХ в настоящее время исключена (это можно видеть в прилагаемом образце).

В 8, 9, 10 и 11 графах указывается возможность безвозмездного использования копий программы для слепых и физически недееспособных лиц, реквизиты банковского счета заявителя, адрес для переписки, удостоверяется правильность понимания заявителем содержания Формы ТХ и ставится его подпись.

Причина столь подробного рассмотрения данной формы состоит в том, что ее неправильное заполнение и неполный комплект необходимых материалов влекут за собой отказ в регистрации, а за «сознательное искажение или предоставление неверных сведений при заполнении заявки» заявитель может быть оштрафован на сумму до двух с половиной тысяч американских долларов.

Еще одним важным вопросом при регистрации являются требования к образцу программы. Для регистрации программ, не содержащих коммерческой тайны, необходимо представить первые и последние двадцать пять страниц визуально воспринимаемого описания программы на том или ином языке программирования или, если вся программа занимает менее пятидесяти страниц, полное ее описание, а также страницу, содержащую знак авторских прав. Для программ, созданных в результате творческой переработки, необходимо приложить любые пятьдесят страниц описания, содержащие внесенные изменения. Если программа разработана с использованием системы HyperCard, устанавливаются те же требования по количеству страниц описания, что и для программ, написанных с помощью кодов.

При наличии в программе коммерческих секретов к заявке прилагается страница, содержащая знак авторских прав, а также:

а) для вновь созданной программы — первые и последние двадцать пять страниц описания программы на языке программирования, при этом фрагменты, содержащие коммерческую тайну, изымаются. Однако такие фрагменты по размеру не должны превышать пятидесяти процентов высылаемого описания программы. Возможно отправление первых и последних десяти страниц, не содержащих секретов. Если описание программы на языке программирования занимает менее пятидесяти страниц, высылается полное описание с изъятыми фрагментами с коммерческой тайной.

б) для творчески переработанной программы, если изменения содержатся в первых и последних двадцати пяти страницах описания, порядок оформления образцов такой же, как и для вновь созданной программы. В противном случае прилагаются любые двадцать страниц с внесенными изменениями без коммерческих секретов или пятьдесят страниц с изменениями и изъятыми фрагментами, содержащими такие секреты.

Особое место среди охраняемых и подлежащих регистрации элементов программы занимают изображения на экране. На основе обобщения судебной практики Отделом авторских прав в 1988 году было издано постановление, по которому этот компонент программы входит в охраняемые объекты и не требует отдельной регистрации или отдельного упоминания в заявке. Это решение относится и к графическим изображениям видеоигр. При этом в общем случае, когда регистрируется программа в целом, для регистрации необязательно прилагать образцы графики. Если же в заявке в качестве регистрируемого объекта указывается именно экранное изображение, такой образец обязателен. Ниже приводятся примеры заполнения графы 2 Формы ТХ при регистрации изображений.

| Пример заполнения | Требования к образцу экранного изображения |
|---|---|
| 1. Программа в целом. | Прилагать необязательно |
| 2. Программа в целом, включая тексты на экране. | |
| Программа в целом, включая аудиовизуальные материалы. | Прилагать обязательно |
| Программа в целом, включая изображения. | |

При этом не охраняются изображения меню, структура, формат изображений и тому подобное. Образцы должны представлять собой распечатки, фотографии или рисунки, ясно определяющие регистрируемый материал.

Программы, впервые опубликованные до 1.03.89 г., должны содержать знак авторских прав. Для программ, публикация которых имела место после этой даты, наличие знака необязательно, однако настоятельно рекомендуется, так как дает преимущества при рассмотрении исков о нарушении прав автора в суде (в частности, ответчик не сможет ссылаться на «неумышленное нарушение», которое влечет снижение размеров вины и возмещения убытков владельцу авторских прав). Знак представляет собой символ ©, надпись «Copyright» или аббревиатуру «Copr», дату первого опубликования программы и имя владельца авторских прав. Если программа представлена на носителях (магнитная лента, дискета), знак авторских прав должен воспроизводиться рядом с названием работы при ее распечатке или же отображаться на дисплее в процессе работы с программой. Допускается воспроизведение знака непосредственно на носителе или его упаковке. ■

# FORM TX
UNITED STATES COPYRIGHT OFFICE

REGISTRATION NUMBER

TX TXU

EFFECTIVE DATE OF REGISTRATION

Month Day Year

**DO NOT WRITE ABOVE THIS LINE. IF YOU NEED MORE SPACE, USE A SEPARATE CONTINUATION SHEET.**

**1** **TITLE OF THIS WORK ▼**

**PREVIOUS OR ALTERNATIVE TITLES ▼**

**PUBLICATION AS A CONTRIBUTION** If this work was published as a contribution to a periodical, serial, or collection, give information about the collective work in which the contribution appeared. **Title of Collective Work ▼**

If published in a periodical or serial give: **Volume ▼** **Number ▼** **Issue Date ▼** **On Pages ▼**

**2** a **NAME OF AUTHOR ▼**

**DATES OF BIRTH AND DEATH**
Year Born ▼ Year Died ▼

Was this contribution to the work a "work made for hire"?
☐ Yes
☐ No

**AUTHOR'S NATIONALITY OR DOMICILE**
Name of Country
OR { Citizen of ▶ _____
Domiciled in ▶ _____

**WAS THIS AUTHOR'S CONTRIBUTION TO THE WORK**
Anonymous? ☐ Yes ☐ No
Pseudonymous? ☐ Yes ☐ No
If the answer to either of these questions is "Yes," see detailed instructions.

**NATURE OF AUTHORSHIP** Briefly describe nature of the material created by this author in which copyright is claimed. ▼

**NOTE**
Under the law, the "author" of a "work made for hire" is generally the employer, not the employee (see instructions). For any part of this work that was "made for hire" check "Yes" in the space provided, give the employer (or other person for whom the work was prepared) as "Author" of that part, and leave the space for dates of birth and death blank

b **NAME OF AUTHOR ▼**

**DATES OF BIRTH AND DEATH**
Year Born ▼ Year Died ▼

Was this contribution to the work a "work made for hire"?
☐ Yes
☐ No

**AUTHOR'S NATIONALITY OR DOMICILE**
Name of country
OR { Citizen of ▶ _____
Domiciled in ▶ _____

**WAS THIS AUTHOR'S CONTRIBUTION TO THE WORK**
Anonymous? ☐ Yes ☐ No
Pseudonymous? ☐ Yes ☐ No
If the answer to either of these questions is "Yes," see detailed instructions.

**NATURE OF AUTHORSHIP** Briefly describe nature of the material created by this author in which copyright is claimed. ▼

c **NAME OF AUTHOR ▼**

**DATES OF BIRTH AND DEATH**
Year Born ▼ Year Died ▼

Was this contribution to the work a "work made for hire"?
☐ Yes
☐ No

**AUTHOR'S NATIONALITY OR DOMICILE**
Name of Country
OR { Citizen of ▶ _____
Domiciled in ▶ _____

**WAS THIS AUTHOR'S CONTRIBUTION TO THE WORK**
Anonymous? ☐ Yes ☐ No
Pseudonymous? ☐ Yes ☐ No
If the answer to either of these questions is "Yes," see detailed instructions.

**NATURE OF AUTHORSHIP** Briefly describe nature of the material created by this author in which copyright is claimed. ▼

**3** a **YEAR IN WHICH CREATION OF THIS WORK WAS COMPLETED** **This information must be given in all cases.**
_____ ◀ Year

b **DATE AND NATION OF FIRST PUBLICATION OF THIS PARTICULAR WORK**
**Complete this information ONLY if this work has been published.**
Month ▶ _____ Day ▶ _____ Year ▶ _____
_____ ◀ Nation

**4** **COPYRIGHT CLAIMANT(S)** Name and address must be given even if the claimant is the same as the author given in space 2.▼

See instructions before completing this space

**TRANSFER** If the claimant(s) named here in space 4 are different from the author(s) named in space 2, give a brief statement of how the claimant(s) obtained ownership of the copyright.▼

**DO NOT WRITE HERE OFFICE USE ONLY**
APPLICATION RECEIVED
ONE DEPOSIT RECEIVED
TWO DEPOSITS RECEIVED
REMITTANCE NUMBER AND DATE

EXAMINED BY

CHECKED BY

CORRESPONDENCE
Yes

**FORM TX**

FOR COPYRIGHT OFFICE USE ONLY

**DO NOT WRITE ABOVE THIS LINE. IF YOU NEED MORE SPACE, USE A SEPARATE CONTINUATION SHEET.**

**PREVIOUS REGISTRATION** Has registration for this work, or for an earlier version of this work, already been made in the Copyright Office? 5

☐ **Yes** ☐ **No** If your answer is "Yes," why is another registration being sought? (Check appropriate box) ▼

☐ This is the first published edition of a work previously registered in unpublished form.

☐ This is the first application submitted by this author as copyright claimant.

☐ This is a changed version of the work, as shown by space 6 on this application.

If your answer is "Yes," give: **Previous Registration Number** ▼ **Year of Registration** ▼

**DERIVATIVE WORK OR COMPILATION** Complete both space 6a & 6b for a derivative work; complete only 6b for a compilation. 6

**a. Preexisting Material** Identify any preexisting work or works that this work is based on or incorporates. ▼

**b. Material Added to This Work** Give a brief, general statement of the material that has been added to this work and in which copyright is claimed. ▼

See instructions before completing this space.

## —space deleted— 7

**REPRODUCTION FOR USE OF BLIND OR PHYSICALLY HANDICAPPED INDIVIDUALS** A signature on this form at space 10, and a check in one of the boxes here in space 8, constitutes a non-exclusive grant of permission to the Library of Congress to reproduce and distribute solely for the blind and physically handicapped and under the conditions and limitations prescribed by the regulations of the Copyright Office: (1) copies of the work identified in space 1 of this application in Braille (or similar tactile symbols); or (2) phonorecords embodying a fixation of a reading of that work; or (3) both. 8

a ☐ Copies and Phonorecords b ☐ Copies Only c ☐ Phonorecords Only

See instructions

**DEPOSIT ACCOUNT** If the registration fee is to be charged to a Deposit Account established in the Copyright Office, give name and number of Account. 9

**Name** ▼ **Account Number** ▼

**CORRESPONDENCE** Give name and address to which correspondence about this application should be sent. Name/Address/Apt/City/State/Zip ▼

Area Code & Telephone Number ▶

Be sure to give your daytime phone number.

**CERTIFICATION*** I, the undersigned, hereby certify that I am the 10

Check one ▶
- ☐ author
- ☐ other copyright claimant
- ☐ owner of exclusive right(s)
- ☐ authorized agent of _____

Name of author or other copyright claimant, or owner of exclusive right(s) ▲

of the work identified in this application and that the statements made by me in this application are correct to the best of my knowledge.

**Typed or printed name and date** ▼ If this application gives a date of publication in space 3, do not sign and submit it before that date.

**date** ▶

**Handwritten signature (X)** ▼

**MAIL CERTIFICATE TO** 11

Certificate will be mailed in window envelope

Name ▼

Number/Street/Apartment Number ▼

City/State/ZIP ▼

**YOU MUST:**
- Complete all necessary spaces
- Sign your application in space 10

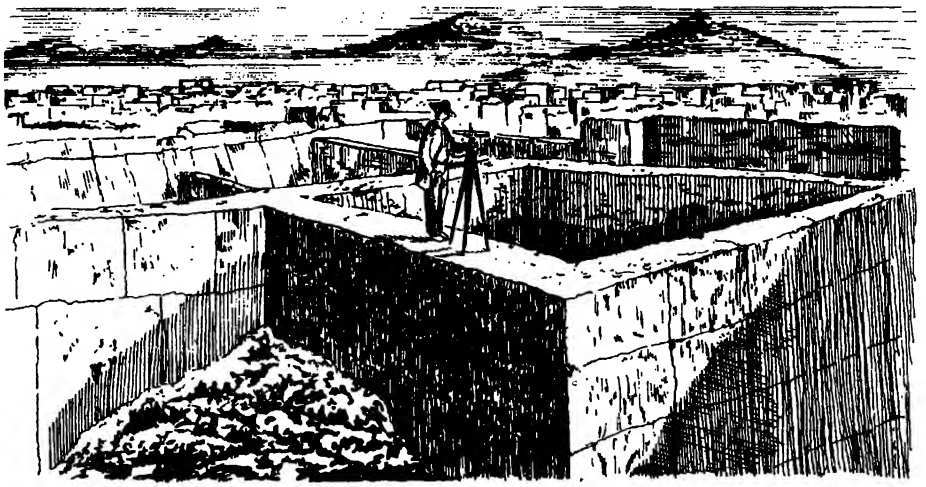
**SEND ALL 3 ELEMENTS IN THE SAME PACKAGE:**
1. Application form
2. Non-refundable $10 filing fee in check or money order payable to *Register of Copyrights*
3. Deposit material

**MAIL TO:**
Register of Copyrights
Library of Congress
Washington, D.C. 20559

* 17 U.S.C. § 506(e): Any person who knowingly makes a false representation of a material fact in the application for copyright registration provided for by section 409, or in any written statement filed in connection with the application, shall be fined not more than $2,500.

# Подземный город (по мотивам программы The Eye of Beholder)

*Литературная обработка Ю. Матвеева*



**«ИНФОРКОМ» представляет**

*Раздел, освещающий игровые программы, понравился многим читателям журнала. Каждый день почта приносит просьбы рассмотреть те или иные программы. Читатели хотят, чтобы им раскрыли секреты, которые позволят до конца пройти полюбившуюся игру. Мы очень хорошо понимаем их мотивы, но, к сожалению, существуют и проблемы. Ведь есть в «расколотой» программе что-то от наполовину переваренного обеда, не правда ли?*

*Экспериментируя в этом направлении, «ИНФОРКОМ» пришел к новому жанру — мы называем его «компьютерная новелла». Главная цель жанра — дать увлекательное описание игры, содержащее в завуалированном виде достаточное количество подсказок и рекомендаций для ее успешного завершения, что не только не снижает интереса к игре, но и наоборот способствует ее популярности.*

*С этим направлением нашей деятельности уже давно знакомы те пользователи персональных компьютеров системы «Sinclair», которые регулярно читают «ZX-РЕВЮ» (фирменное издание «ИНФОРКОМа»). Сегодня же, идя навстречу просьбам читателей «Монитора», мы публикуем первую новеллу для пользователей IBM совместимых машин.*

*Свои отзывы, предложения и пожелания просим направлять по адресу: 121019 Москва, Г-19, а/я 16, «ИНФОРКОМ», а также в редакцию журнала «Монитор».*

...Теперь я уже окончательно пришел в себя и могу спокойно рассказать о том, что произошло. Лишь один вопрос тревожит меня: можно ли в это поверить? Я готов согласиться с тем, что людям иногда снятся такие сны, когда реальность и миф, переплетаясь, становятся чем-то третьим, для чего трудно найти название. Но это все было! Я точно знаю, что это не сон.

Мне трудно объяснить, как я оказался там. Я просто закрыл глаза и увидел ИХ. Только потом я понял, что стал свидетелем таинственной встречи магов — правителей средневекового города Уотердип.

В первые мгновения мое внимание привлекли контуры старинной высокой башни, где проходила встреча. Я парил в воздухе и не ощущал своего тела. В окне, под самой крышей башни, я заметил неясные очертания человеческих фигур в длинных черных мантиях. Легкий ветерок доносил обрывки фраз, из которых я понял, что над городом нависла смертельная опасность и никакие магические заклинания и древнее волшебство не могут противостоять злу.

Тогда кто-то, видимо, самый главный маг, произнес слова, с которыми все согласились:

– Нам нужны добровольцы, способные умом и силой победить зло, пришедшее на нашу землю. Я уверен, что в Уотердипе найдутся смельчаки, готовые пожертвовать собой во имя всеобщего благоденствия. И это наш единственный шанс!

Не могу объяснить как, но совершенно неожиданно для себя я услышал другой, скрипящий и неприятный голос, который шел извне:

– Они думают, что это им поможет...

Все еще паря в воздухе, я огляделся, но никого не увидел.

Совет магов принял решение, и сразу весь Уотердип облетела весть о найме добровольцев, готовых служить во славу родного города.

На следующее утро у королевского дворца собралась порядочная толпа любопытных, пришедших посмотреть на смельчаков, откликнувшихся на призыв магов. Время шло, а желающих помочь городу не было. Верховный правитель скучал в тронном зале, изредка поглядывая на пеструю толпу горожан у дворцовых ворот.

Я парил между небом и землей, наблюдая сверху за всем происходящим. Я чувствовал, что могу помочь незнакомому мне городу во имя добра. Недаром мне было суждено, минуя пространство и время, оказаться в самом центре этих загадочных событий. Мое желание увидеть смелых и сильных добровольцев было настолько велико, что я реально представил себе, как они выходят из толпы зевак и идут к воротам дворца. Велико же было мое удивление, когда вся толпа неожиданно вздохнула и расступилась, — от нее отделились четыре маленькие фигуры и не спеша направились ко входу во дворец. Сам того не ведая, я, похоже, материализовал свою фантазию.

Перенесясь в тронный зал вслед за своими героями, я не успел разглядеть их, когда они, выслушав короткое напутствие верховного правителя, двинулись к выходу.

Кто эти люди? Плод моей фантазии или реально существующие жители города? На этот вопрос я не нашел ответа. Сначала я представлял их лица с трудом, но потом из глубин подсознания выплыли яркие, запоминающиеся образы. Может быть, я когда-то встречал их в своей жизни или эти лица были из той области памяти, которую обычно называют памятью предков. Мне трудно объяснить свои ощущения. Однако, когда я вплотную увидел их всех четверых, они оказались именно такими, какими я их представлял. Я допускаю, что, будь на моем месте другой человек, в его сознании родились бы совсем другие образы.

Первого, огромного бородатого рейнджера с обветренным лицом и черными глазами, звали Аллан. Я внутренне ощущал его силу, ум, сообразительность, хороший уровень боевой подготовки. Это был воин высокого класса и с большим опытом.

Второй, приземистый рубаха-парень, с правильными чертами лица, тоже был прекрасным смелым воином. Его звали Томас. Или мне хотелось, чтобы его так звали?

Третьей была женщина по имени Оливия. Изящная, строгой красоты, она не была похожа на колдунью, но я-то знал, что это не так. Оливия хорошо владела магическими заклинаниями, которые имели боевую направленность. У нее уже имелся опыт сражения с разного рода нечистью и черной магией. Возможно, ей далеко было до совершенства, но ученицей она была прекрасной.

Четвертого — последнего участника — звали Джефф. Это был уже пожилой, но еще крепкий священник. Густая борода скрывала пол-лица, а из-под низкого открытого лба смотрели прищуренные глаза. Джефф умело пользовался священными молитвами для лечения раненных в бою и защиты от всякого рода напастей. Он уже ходил под знаменами войск с крестом в руке и Богом в душе. Однако, когда это было необходимо, Джефф мог поработать и чугунной палицей с острыми шипами.

Я следовал за отрядом с самого начала их путешествия и был их частью. Я ощущал себя внутри каждого и в то же время был вне их. Невидимые нити связали наши судьбы, и тем самым я стал участником событий, развернувшихся в подземелье города.

По словам Верховного правителя, зло и опасность исходили оттуда. Вся группа, снарядившись самым необходимым, отправилась к железной решетчатой двери, находившейся на отвесном склоне гранитной мостовой неподалеку от дворца. Ржавая дверь со скрипом распахнулась, и они вошли внутрь. В первом ряду шли Аллан и Томас, а на шаг сзади — Оливия и Джефф. Этот порядок они приняли без обсуждения, и он надолго стал их боевым порядком.

Не успели они пройти несколько метров, как услышали нарастающий гул. Крик Оливии совпал с оглушительными раскатами грома. Это было похоже на землетрясение. Позади них с потолка недалеко от входа посыпались песок и огромные валуны, которые, падая, расплескивали по стенам серо-зеленую жижу, покрывающую тоннель. Через секунду выход был плотно завален камнями. И опять, как в первый раз, я услышал знакомый мне потусторонний голос:

– Теперь они в моих в руках!

– Ничего, — прошептал Аллан, — мы найдем другой выход.

– А что, если его вообще не существует? — Томас стоял рядом, опершись спиной на холодную каменную стену тоннеля. Оливия смотрела на завал и молчала. Пожилой Джефф подтянул за плечами вещевой мешок.

– Мой дед, когда я еще был птенцом, рассказывал мне об этом подземелье. В его времена ходили слухи, что где-то был еще один выход, который, возможно, уже давно занесло песком и сровняло с землей.

– Все равно придется идти вперед, — отозвался Аллан. — Быть может, кто-нибудь нам подскажет, как выбраться отсюда.

Восстановив боевой порядок, они двинулись навстречу неизведанному.

Через некоторое время группа наткнулась на большую железную дверь, которую Джефф без труда открыл рычагом, находящимся рядом. Пройдя чуть дальше, они заметили еще одну точно такую же дверь, которая при приближении распахнулась сама, но неожиданно захлопнулась, когда друзья подошли к ней вплотную. Аллан внимательно осмотрел коридор. Потом, не говоря ни слова, поднял валявшийся под ногами камень и, вернувшись назад, положил его на круглую крышку какого-то приспособления, крепко встроенного в пол. Дверь опять распахнулась и больше не закрывалась. Все четверо прошли дальше. Под ногами хлюпала грязь, в воздухе пахло сыростью и плесенью. Преодолев несколько пролетов, они вышли в небольшое помещение. Прямо напротив по центру стены находилась ржавая металлическая дверь. Справа от нее на общем фоне отчетливо выделялась закамуфлированная под кирпич небольшая кнопка. Томас перешагнул через человеческие кости, которые лежали тут же, и дотянулся до кнопки. Дверь скрипнула и подалась, но они не успели и шагу шаг-

нуть, как из открывшегося прохода выползла болотного цвета гигантская пиявка. Извиваясь своим жирным телом, нечисть периодически раскрывала зубастую пасть, пытаясь вцепиться кому-нибудь в ногу.

Аллан, выхватив кинжал, в два удара прикончил тварь. Но тут из-за угла появилась еще одна, которой удалось схватить его за ногу. В этой ситуации не оплошал коренастый Томас. Ударив своим тяжелым мечом по жирному телу, он разрубил пиявку пополам. Аллан с посеревшим от боли лицом благодарно посмотрел на Томаса. Первый бой они выиграли, и это было хорошим предзнаменованием. А чуть дальше их ждал сюрприз: Джефф высмотрел где-то на стене огромный кирпичный блок, слегка выделявшийся среди остальных.

– Ну-ка, Томас, попробуй вдавить его в стену, — задумчиво произнес Джефф, поглаживая роскошные усы.

Однако больших усилий прикладывать не пришлось. Едва только Томас дотронулся до подозрительного блока, как вся стена слева от них прямо на глазах растворилась в воздухе. Джефф несколько раз перекрестился, а Оливия, открыв свою книгу магических заклинаний, что-то тихо прошептала.

– Все нормально, — услышали они голос Аллана, который, осмотрев проход, махнул им рукой. Но, не пройдя и десяти метров, они снова столкнулись с подземными обитателями. Это были какие-то карлики, очень агрессивно настроенные, которые кинулись на Аллана и Томаса с ножами в руках. Однако они оказались слабыми и никудышными бойцами: достаточно было одного удара, чтобы отправить в мир иной их души. Томас и Аллан довольно быстро уничтожили почти целый отряд карликов. Не оставались в стороне и Оливия с Джеффом: они обстреливали карликов камнями, собранными в коридорах подземелья.

Когда с карликами было покончено, Аллан, вытирая со лба пот, повернулся к Джеффу:

– Кто это был? — спросил он.

– Похоже, что это кобольды, — отозвался Джефф. — Злые кровожадные существа, питающиеся человеческим мясом.

– Кстати, — заметил Томас, — неплохо было бы перекусить и отдохнуть.

– Уж не собираешься ли ты пообедать этими людоедами? — засмеялась Оливия.

– Нет, конечно, — нахмурился Томас, стаскивая с плеч вещевой мешок. — У меня кое-что повкуснее найдется.

– Ну, хорошо, — потянулся Аллан, — привал, так привал.

Два часа пролетели незаметно. Оливия за это время успела переписать со свитков, найденных в подземелье, несколько новых заклинаний в свою магическую книгу. Кое-что она выучила наизусть. Джефф долго молился, призывая Господа Бога помочь странникам победить в следующих битвах. Томас и Аллан приводили в порядок оружие. После битвы с кобольдами они пополнили запас стрел и кинжалов. Через некоторое время друзья снова отправились в путь. Им еще раз встретилась небольшая группа кобольдов, которая была быстро разгромлена. После этой битвы особых препятствий больше не возникало.

Вскоре они нашли лестницу, ведущую на нижние уровни подземелья.

Второй уровень был гораздо больше по размерам и казался бесконечным. Здесь они нашли много полезного: Томас облачился в старинную кольчугу, Аллан уже примеривал тяжелый и прочный шлем с забралом, покрытым шипами, Оливия запаслась пузырьками с какой-то волшебной жидкостью, а Джефф приспособил на поясе отлично сделанную пращу. Несколько раз Томасу и Аллану пришлось попотеть, открывая заклинившиеся двери. Они распознавали такую дверь по щели у пола. Чудеса начались позже, когда все обнаружили, что пришли на то же место, где уже были. Когда ситуация повторилась дважды, у Томаса сдали нервы:

– Мы блуждаем здесь уже второй час, а оказывается бродим по кругу?

– Не горячись, — Джефф озадаченно смотрел на компас. — Следите все за стрелкой.

Они шли прямо и не меняли направления, однако в какое-то мгновение стрелка вместо юга сразу повернулась на север.

– Нас развернули, — сказала Оливия. — Мы этого не заметили, но это так. — Она осмотрела пол и стены коридора.

– Чертовы волшебники! — Томас плюнул себе под ноги и рубанул ладонью воздух.

– Нам надо быть внимательнее, — отозвался Джефф, — и мы найдем новую дорогу. Идем!

У Томаса округлились глаза, когда он увидел, как Джефф совершенно неожиданно прошел сквозь стену и исчез.

– Сюда, сюда! — услышали все трое голос Джеффа.

Оказалось, что каменное великолепие одной из стен коридора было иллюзией. Проход был свободен. Томас рвался в бой, желая отомстить неведомому волшебнику за такие мерзкие, по его словам, шутки. Но скоро Джефф нашел ему небольшую работу, которая на время отвлекла воина. За одной решетчатой дверью, плотно запертой, они разглядели потайную кнопку. Дотянуться до нее рукой сквозь решетку не было никакой надежды. Зато Томас, взяв у Джеффа небольшой булыжник, продемонстрировал великолепную точность метания. Пролетев сквозь решетчатую дверь, камень стукнулся о кнопку, и дверь открылась. Побродив еще около часа по лабиринту, они нашли «движущуюся комнату». Так ее назвал Джефф. Здесь он обнаружил кнопку, которая приводила в действие скрытый механизм, разворачивающий комнату по трем различным направлениям. Пройдя наугад несколько пролетов, они неожиданно столкнулись с зомби. Из темноты коридора, выставив вперед руки с пожелтевшими крючковатыми пальцами, к ним шел мертвец. Он был одет в лохмотья, бледное лицо светилось фосфором, а стеклянные глаза смотрели без всякого выражения. Аллан махнул кинжалом. Томас поддержал его справа крепким ударом меча, но зомби устоял на иссохших ногах и вцепился когтями Томасу в грудь. Железная кольчуга затрещала под натиском каменных пальцев. На помощь пришла Оливия, прокричавшая какое-то древнее заклятие. Зомби сразу развернулся и отступил. Пришедший в себя Томас вонзил ему в спину меч. Мертвец упал и стал быстро разлагаться на глазах. В этот момент из темноты появился еще один зомби. Здесь на высоте оказался Аллан, который добил нечисть из загробной жизни после магического заклятия Оливии.

Друзьям пришлось потратить много сил в этой части подземелья, открывшейся после прохода через «движущуюся комнату».

Они несколько раз вступали в схватку с зомби и скелетами, облаченными в воинские доспехи. Джефф высказал предположение, что скелеты — это останки рыцарей, предавших свой народ и перешедших на службу к дьяволу. Несмотря на это, Аллан и Томас не побрезговали подобрать их вооружение после боя. Они нашли и много хорошей свежей еды, неизвестно откуда взявшейся. Аллан оставил себе прекрасную кольчугу и шлем с резным гребнем, а Томас набрал целую коллекцию всякого рода кинжалов. Они вернулись в «движущуюся комнату» в хорошем настроении. Джефф подлечил раны Аллана, и теперь вся группа могла продолжить свой путь.

Где-то в глубине длинного коридора Оливия нашла золотой ключ, который лежал в каменной нише, расположенной по центру стены. Повертев ключ в руках, Джефф сказал:

– Я думаю, этот ключ нам пригодится.

Они вернулись назад и снова вошли в уже знакомую всем комнату. Томас нажал на кнопку, развернув тем самым стены в исходное состояние.

Немного поблуждав по лабиринту, они увидели глубокие ямы, тянущиеся по всей длине коридора. В одну из таких ям чуть было не провалился Томас, который отошел немного вперед. Его вовремя окликнул Джефф:

– Лучше найти хорошую прочную лестницу, чем лечить потом твои переломы.

Спустя несколько минут с помощью найденного золотого ключа они пробрались на третий подземный уровень. Ничего нового для себя они здесь не открыли. Иногда неведомые силы отбрасывали их назад, причем никто этого не замечал, лишь коридор после такого волшебства казался длиннее, чем он был на самом деле. Некоторые двери Джефф открывал с помощью найденных в лабиринте драгоценных камней. Он бережно клал камень в специальный ящик, который находился неподалеку, и дверь, скрипя, открывалась.

Побродив по лабиринту почти три часа, они наткнулись на странный подземный музей.

– Вот это да! — вырвалось у Томаса при виде разных монстров, стоявших неподвижно, словно статуи.

Аллан не отрываясь смотрел на жабоподобное создание ростом с человека, величественно возвышавшееся посредине музея. Рядом находилось несколько точно таких же фигур, но были здесь и другие существа — маленькие, напоминающие человека, но с медвежьей головой.

– Ох, не нравится мне все это, — перекрестился Джефф.

– Скорее пойдемте отсюда, — взмолилась Оливия, отступая к выходу.

– Подождите, — остановил их Аллан, — там что-то лежит! — Он сделал несколько шагов вперед. Прикрывая его, прошел вперед и Томас.

– Отличный щит, — сказал Аллан, увидев лежащий на полу металлический круг с замысловатым рисунком. Он протянул руку, чтобы взять его, но монстры, окружавшие их, вдруг ожили и сразу бросились на нарушителей покоя. Все превратилось в сплошной кошмар. Молитвы Джеффа и заклинания Оливии перемешались с ревом нападавших и криками Аллана и Томаса. Брызги крови разлетались в стороны и окрашивали стены. Из распахнутых пастей жабоподобных текла отвратительная слизь. С этими монстрами справиться было нелегко, ибо они владели черной магией и успешно применяли ее в бою. Друзьям пришлось отступить и занять позицию, прикрытую стенами, чтобы уменьшить число одновременно атакующих монстров. Это была удачная тактика, и они уже применяли ее не раз. В конце концов бой был выигран.

Отдохнув и перекусив, друзья тронулись в путь. Им пришлось повозиться с причудливыми рисунками на стенах, изображавшими очки с одним светящимся глазом. Джефф предложил вставить в каждую пару очков еще по одному глазу, используя найденные драгоценные камни. Так им удалось открыть проход на четвертый уровень.

Здесь Аллан и Томас расправились с гигантским пауком, который внезапно напал сзади. Джефф, заподозрив неладное, предложил всем вернуться уровнем выше. Поднявшись наверх, они вновь спустились на четвертый уровень, но уже по другой лестнице. Когда Томас спросил о смысле этих странных перемещений, Джефф в ответ пробормотал что-то невнятное про пауков. Он оказался прав. Та часть лабиринта, куда они пошли сначала, буквально кишела гигантскими пауками. Только об этом друзья узнали позже, а сначала они заметили гнома, сидевшего на каменном полу. Одной рукой он прикрывал рану, из которой сочилась кровь. Увидев приближающихся воинов, гном приподнял голову и что-то прохрипел.

– Ждите меня здесь, — сказал Джефф и направился к раненому.

Аллан, Томас и Оливия остались в стороне, наблюдая, как опытный лекарь залечивает рану. Никто не сомневался, что Джеффу удастся привести гнома в чувство и спасти от неминуемой смерти. Используя священные заговоры и какие-то примочки, Джефф довольно быстро вылечил раненого. Когда гном окончательно пришел в себя, он вежливо поблагодарил своих спасителей. Потом он поведал, что его зовут Тагор, что он участвовал в неравной битве гномов с эльфами и был ранен, когда их отряд отступал.

– В знак благодарности, — сказал он заканчивая, — я готов присоединиться к вам и следовать за вами хоть на край света. — Приложив руку к груди, Тагор низко поклонился.

– Хорошо, — сказал Джефф, — но у нас будет трудный путь.

Аллан предложил Тагору добротный лук и стрелы, но тот вежливо отклонил оружие. — Нет для гнома более верного спутника в бою, чем надежный топор, а лук — это оружие коварных эльфов.

К счастью, в багаже героев нашелся старый, прошедший многие битвы тяжелый топор.

– Будешь прикрывать нас сзади, — деловито пояснил Томас, слегка сбитый с толку размерами новобранца.

Томас напрасно сомневался в способностях Тагора. Тот оказался неплохим воином, а топор в его руках, казалось, терял свой вес. Все убедились в этом, когда на отряд напали гигантские пауки. Пауки не были грозными соперниками, однако их яд был очень опасен. В одном из боев от ядовитого укуса паука пострадал Джефф. Когда сражение окончилось, все обра-

тили внимание на священника, который стал отставать от группы, все чаще и чаще прислоняясь к стене, чтобы отдышаться. Его лоб покрылся крупными каплями пота, а сухие потрескавшиеся губы жадно хватали воздух.

– Он отравлен, — сказал Тагор. — Нам нужно волшебное противоядие — напиток, который вернет его к жизни.

– Не этот ли? — спросил Аллан, доставая из вещевого мешка сосуд с голубой жидкостью. — Я нашел его здесь, на этом уровне.

– Именно он! — воскликнул Тагор.

– У меня есть еще три таких сосуда, — поделился Аллан с Оливией, глядя на то, как седобородый гном ловко вливает жидкость в приоткрытый рот Джеффа. — Нам надо их раздать, чтобы у каждого было под рукой противоядие. — Все были приятно удивлены, когда увидели, как порозовело лицо священника и он быстро пришел в себя. Теперь они снова могли продолжить путь.

Тагор сообщил, что отряд гномов находится уровнем ниже, и если они туда доберутся, возможно, узнают что-то новое и получат подмогу.

А пройти ниже оказалось не так уж просто: проходы открывались и закрывались по непонятным причинам, сквозь паутину приходилось буквально продираться, используя мечи и кинжалы, да к тому же, все время следуя на север, они вдруг вышли в южную часть подземелья. Кое-как разобравшись с местной географией, отряд обнаружил большую, с крупным орнаментом дверь. Это был магический портал, который мог переносить тела странников на расстояние. По рисунку орнамента они догадались, что воспользоваться порталом можно, взяв какую-либо вещь, изображения которой недостает для логического завершения композиции. Для этого портала необходимы были каменный кинжал и скипетр.

Оказавшись на пятом подземном уровне, они долго плутали между иллюзорными стенами по пустынным коридорам, пока не встретили отряд гномов.

Их предводитель по имени Армун рассказал историю про то, как коварные эльфы тяжело ранили короля гномов и похитили молодого принца. Многие гномы видели, как отряды эльфов вела в бой злая женщина по имени Шиндия, которая, видимо, имела отношение к похищению принца.

– Если вы на нашей стороне, — сказал Армун, — помогите нам найти принца, ибо наш отряд сейчас не в силах идти на поиски. — Армун вопросительно смотрел на всех пятерых.

– Мы согласны помочь вам, — за всех ответил Аллан.

Предводитель гномов заметно повеселел и поделился секретами использования порталов, а также рассказал про оракула, который может поведать о магических свойствах предметов.

– Для этого, — сказал Армун, — вам нужно будет найти магический шар, с помощью которого активизируется оракул.

Только сейчас друзья поняли, о каком оракуле идет речь: они видели его уровнем выше, но не обратили особого внимания. Теперь кое-что прояснилось, однако где находится магический шар, Армун не знал.

Из-за спины Армуна появился гном, который попросился в отряд людей:

– Меня зовут Дохрум, — сказал он, — и я хочу помочь вам в поисках принца.

Теперь их стало шестеро. Они получили в подарок от гномов каменный медальон и много хорошей еды. Главный целитель подземных жителей помог всем подлечить раны, а после добавил:

– Я могу не только лечить, но и оживлять людей. Если вам будет что-то нужно, всегда рад помочь.

Попрощавшись с гномами, они вновь двинулись в путь по бесконечным коридорам лабиринта. Подземный город впечатлял своими размерами. Сколько прошло времени, никто точно не знал. Они окрепли в боях, стали мудрее и сильнее. Собранное в пути оружие и предметы снаряжения разумно использовались. Джефф ловко вычислял хитрые ловушки, которые подстраивали неведомые силы; Аллан и Томас уверенно действовали в столкновениях с нечистью; Тагор и Дохрум сокрушали противников топором и секирой, а Оливия умело использовала магию.

На шестом уровне группа неожиданно столкнулась лицом к лицу с каким-то высоким человеком, одетым в синюю мантию. Без сомнения, это был маг. Судя по его одежде и украшениям, он принадлежал к богатому сословию. При виде бойцов губы его расплющились в улыбке, и он поднял руку, приказывая остановиться.

– Весьма наслышан о вас, — начал маг, окинув всех непроницаемым взглядом, — и я рад, что вы живы и здоровы.

В ответ никто не произнес ни слова. Тогда он продолжил:

– Я знаю, что вы ищете главного злодея, посылающего скверну на ваш город. Что ж, Ксанатар действительно планирует атаковать Уотердип, как только тот будет ослаблен. И я знаю: Ксанатар победит. Против него очень трудно воевать, но и на Ксанатара найдется управа: в битве с ним можно очень удачно использовать специальный магический жезл, который есть у гномов. Но они о нем почти ничего не знают. — Маг сделал паузу, наблюдая за реакцией слушателей. Все внимательно смотрели на мага и молчали. Лишь Томас попытался было задать какой-то вопрос, но Джефф одернул его.

– Когда у меня будет жезл гномов, — сверкнул глазами маг, — я атакую Ксанатара и скину его с пьедестала, как только он захватит город.

Друзья еще только пытались осмыслить сказанное магом, когда тот, заканчивая свою речь, произнес:

– А сейчас, конечно, я убью вас, несчастные спасители Уотердипа...

Не договорив последнего слова, маг сильно чем-то ударил Аллана, стоявшего к нему ближе всех. Томас выхватил меч, но только успел замахнуться, и сразу резко осел после какого-то магического заклинания, которое прошептал злодей. Сзади засвистели стрелы, выпущенные Джеффом. Оправившись от удара, Аллан крепко рубанул мечом, но черная колдовская сила отвела удар. К драке подключились гномы. Оливия с помощью магических заклинаний разрушала чары злодея, делая его все более уязвимым. Бой был трудным, но длился всего несколько секунд, которые показались вечностью. Осыпанный стрелами, пропустивший несколько ударов Аллана и гномов и добитый заклятьями Оливии, чародей отступил и рухнул на пол. Это была победа. Тело мага растворилось на глазах, а оставшееся после него оружие подобрал Тагор.

Томас долго не мог прийти в себя после этой схватки, но благодаря Джеффу для него все закончилось благополучно.

Где-то в глубинах подземелья им несколько раз пришлось уворачиваться от дротиков с ядовитыми наконечниками, которые вылетали неизвестно откуда. Аллан собирал эти дротики, надеясь, что они пригодятся в будущем.

Иногда им приходилось прыгать в огромные, но не очень глубокие колодцы, чтобы взять ключи для выхода на следующий уровень. Часто на них нападали воинственно настроенные птицы, которые оказались очень серьезными противниками. После одного из боев гномы подобрали в глубине коридора яйца этих птиц, которые они считали деликатесом.

Добравшись до седьмого уровня, группа сделала передышку, чтобы подлечиться и перекусить. Тагор и Дохрум угощали всех птичьими яйцами и сами их усердно поедали.

– Надо как следует подкрепиться, — советовал Тагор, — ведь с этого уровня начинаются владения эльфов, и нам будет очень нелегко.

– А как далеко распространяются их владения? — поинтересовался Джефф.

– О, — протянул Дохрум, — надо учиться считать до трех, чтобы понять это.

Все засмеялись, а Томас, похлопав гнома по плечу, сказал:

– Ничего, научимся...

– Когда обойдем их владения, — усмехнулся Джефф.

Ужин, однако, был омрачен встречей с группой эльфов, которая патрулировала свою территорию. Джефф хотел вступить с ними в переговоры, но Аллан и Томас уже схватились за мечи. Эльфы также обнажили оружие. Но обстановку разрядил Тагор, отдав все оставшиеся в запасе яйца птиц. Эльфы с миром удалились.

– Ну что ж, друзья, в путь! — призвал всех Джефф.

На этом уровне вместе с эльфами обитали еще и скелеты-воины, которые оказались гораздо сильнее своих предшественников, встречавшихся раньше. Это были действительно исчадия ада, кровожадные и опасные. В одном из районов лабиринта были обнаружены целые поселения отвратительных скелетов. Но силой и хитростью группе удалось победить стаю монстров.

Потом друзья попали в комнату, в которой было расположено сразу пять магических порталов, а чуть дальше находилась лестница, ведущая на восьмой уровень. Трудная дорога, видимо, близилась к концу, но принца они пока не нашли. Зато пришлось повстречаться с человеко-пауком и огнедышащими пастями. А уж когда они прошли через какой-то зал на девятом уровне, после чего неведомые силы лишили каждого некоторых предметов снаряжения, негодованию их не было предела.

– Зал воров, — прошептал Дохрум. — Здесь всегда что-то крадут.

Иногда их атаковали со всех сторон гигантские насекомые, которые могли мгновенно растворить ядовитой слюной металлические предметы. Бои шли почти без передышки, и только когда они с помощью трех предметов вооружения открыли дверь на десятый уровень, удалось спокойно отдохнуть несколько часов. Они находились в старом заброшенном подземелье, которое сильно заросло мхом и покрылось плесенью.

Друзья долго бродили по подземелью, пока не нашли потайную кнопку, ключ и дверь. После того как Джеффф открыл ее, все увидели женщину. Тагор и Дохрум с перекошенными лицами бросились на нее, но Аллан их остановил.

– Это Шиндия, — перекрикивали друг друга гномы, размахивая руками. — Она руководила эльфами в бою с нашим отрядом, когда похитили принца.

– Не убивайте меня, — взмолилась Шиндия, — падая на колени. — Я могу вам помочь. Я все расскажу! Все! — И она скороговоркой принялась объяснять, что эльфов и гномов столкнул лбами Ксанатар, принца похитили тоже с подачи этого злодея, что она выполняла его приказы и теперь кается.

Немного успокоившись, Шиндия попыталась улыбнуться, но у нее это плохо получилось.

– Я знаю, где находится эликсир, которым можно вылечить короля гномов.

– Где? — выкрикнул Дохрум.

– Уровнем ниже, в комнате рычагов, — ответила Шиндия. — Ой, кто это?

Все обернулись назад, куда показывала рукой женщина, но когда, не заметив ничего подозрительного, повернулись обратно, Шиндии пропал и след.

– Чертовка, — вырвалось у Джеффа.

Гномы, кипя от ярости, топали ногами и били кулаками воздух.

– Идем, — вздохнул Аллан.

Вскоре хмурые лица героев посветлели: они нашли и освободили закованного в цепи Кейргара, так звали принца гномов.

Через порталы они вернулись на пятый уровень и привели живого и невредимого принца в отряд гномов.

Кейргар на радостях сообщил, что выход из подземелья находится на двенадцатом уровне, в древнем поселении гномов. Там можно найти и Ксанатара. После приятной встречи отряд вновь вернулся на десятый уровень.

Джеффу пришлось составить схему лабиринта, чтобы понять, как пройти ниже. Выяснилось, что с десятого на одиннадцатый уровень не ведет ни одна лестница. Туда пришлось спускаться через отверстие в полу коридора. Здесь их ожидало еще большее число ловушек и неприятных мест. Общими усилиями удалось разобраться с кольцевыми стенами, которые опоясывали центральную колонну, и пройти вглубь. Они сходили сначала на запад, где нашли некоторые предметы снаряжения, а потом на север, где после боя с какими-то монстрами нашли ту самую комнату рычагов, про которую говорила Шиндия. Здесь действительно оказалось лекарство для короля гномов. В этой же части лабиринта они обнаружили странный предмет, который потом помог пройти на уровень, где обитал Ксанатар. На востоке, после долгого поиска потайных кнопок, они нашли еще много разных полезных предметов. Пришло время вернуться к гномам и передать им лекарство для короля, однако Оливия предложила сначала посетить четвертый уровень, где можно было определить свойства предметов с помощью оракула, благо магический шар у них уже был. Так они и сделали.

После проверки от некоторых заколдованных нечистой силой предметов пришлось отказаться. Когда они вернулись к гномам, те сначала недоверчиво отнеслись к найденному лекарству, но зато потом, увидев, что их король очнулся и сам встал на ноги, принялись благодарить спасителей. Король торжественно вручил Аллану магический жезл. Это был тот самый жезл, о котором говорил злой маг!

Теперь друзья были полностью вооружены для схватки с самим Ксанатаром.

Простившись с подземными жителями, они вернулись на одиннадцатый уровень.

Душевный трепет охватил меня, когда я понял, что решающая схватка вот-вот состоится. Они шли на юг по лабиринтам, которые постепенно вывели их к порталу, единственному способному перенести отряд в логово злодея Ксанатара. Но перед этим друзьям пришлось драться с существом, которое Оливия назвала «живодером разума». Это существо одним прикосновением могло парализовать весь отряд, а затем постепенно со всеми разделаться. Им удалось победить «живодера» благодаря четким и слаженным действиям всей группы.

Когда они через портал перенеслись на уровень ниже, Оливия предложила хорошо отдохнуть перед последней схваткой. Пока они отдыхали и приводили в порядок снаряжение, я чувствовал, что где-то рядом, за этими изящными мраморными стенами, скрывается Ксанатар.

– Ну и хоромы! — жуя булку и запивая ее молоком, причитал Томас. — Неплохо устроился это дракон.

– А почему ты решил, что он дракон? — спросил Аллан.

Томас в ответ лишь пожал плечами.

Но Ксанатар оказался совсем не похож на дракона. Когда после долгих блужданий по мраморным комнатам они вдруг наткнулись на него, то буквально окаменели. Кто такой был Ксанатар? Мне трудно объяснить. Его шарообразное тело огромных размеров сплошь и рядом было утыкано дюжиной глаз. Некоторые из них располагались на концах небольших отростков, похожих на ножки улиток.

– Я наблюдал за вами с самого начала, — сообщил гремящим голосом монстр. — Вы обошли мои ловушки и преграды. Я понял, что вы хорошие воины, но даже вы не помешаете мне захватить город. Я намного сильнее всех вас. Поэтому предлагаю сдаться, и немедленно...

Аллан и Томас, стоявшие ближе всего к Ксанатару, схватились за рукоятки мечей и прикрылись щитами. Сзади них Джефф поднял крест, а Оливия взяла в руки магическую книгу. Гномы подняли топоры. Все были готовы к смертельной схватке.

– Ах, нет, — захохотал Ксанатар. — Ну, тогда держитесь!...

*...Я открыл глаза и увидел свет. Моя жизнь продолжалась, но уже в другом, прежнем мире. Мои герои остались там, в загадочном средневековом городе Уотердип. И я верю, что они победили. Ведь добро всегда побеждает зло. Я верю в это. Поверьте и Вы.* ■

# Бехолдер — порождение мрака (из досье фирмы TSR, Inc.)

Бехолдер (известный также под названиями *глаз тирана* и *сфера с тысячью глаз*) обитает чаще всего в подземельях, хотя иногда встречается и в диких безлюдных лесах.

Сферообразное тело этого чудовища поддерживается в воздухе посредством левитации и позволяет ему с небольшой скоростью перемещаться в произвольном направлении. В верхней части сферы находятся десять маленьких глаз, а в передней части — огромный одиннадцатый глаз и большой рот с двумя рядами острых, как бритва, зубов.

Бехолдеры могут общаться между собой на своем собственном языке и, как правило, владеют языками других сил мрака (lawful evil).

**Боевые действия.** Тело чудовища защищено толстым хитиновым панцирем, обладающим классом защиты (Armor Class) 0. Малые глаза и центральный глаз также имеют защитное покрытие (классы защит 2 и 7 соответственно)

Каждый из глаз монстра предназначен для выполнения какой-то специальной функции. Как правило, только центральный глаз, а также от одного до четырех малых глаз могут отражать атаку, направленную с фронта. Если второй противник попадает с тыла, монстр направляет на него еще от одного до четырех малых глаз. Это обусловлено анатомическими особенностями строения тела бехолдера и может с успехом использоваться в борьбе с ним.

Каждый из глаз чудовища (за исключением центрального) предназначен для выполнения какого-то одного вида атак. Далее приведены функции малых глаз.

| | |
|---|---|
| 1. | Очаровать гуманоида (charm person). |
| 2. | Очаровать чудовище (charm monster). |
| 3. | Усыпить (Sleep). |
| 4. | Телекинез груза весом 250 фунтов (Telekinese). |
| 5. | Луч превращения в камень (дальность действия 30 ярдов) (Flesh to stone ray). |
| 6. | Распыляющий луч (дальность действия 20 ярдов) (Desintegrate ray). |
| 7. | Деморализовать противника (Fear). |
| 8. | Замедлить действия противника (Slow). |
| 9. | Причинить серьезные повреждения (дальность действия 50 ярдов) (Cause serios wounds). |
| 10. | Луч смерти (дальность действия 40 ярдов) (Death Ray). |
| Центральный глаз. | Луч защиты от магии (дальность действия 140 ярдов) (Anti-magic ray). |

*Луч защиты от магии* охватывает зону в 90 градусов. Все магические способности Вашего героя пропадают до тех пор, пока Вы не покинете зону действия луча. Заклинания, сотворенные внутри зоны или во время прохождения через нее, эффекта не принесут.

Бехолдер довольно успешно защищает свои малые глаза: устойчивость каждого из них составляет от 8 до 12 очков (hit points — hp). Кроме того, поражение малых глаз не входит в снижение общего потенциала чудовища, и потерянные глаза могут отрастать вновь (по одному за неделю). Чудовище способно выдержать две трети общего потенциала повреждений (damage potential), приходящихся на тело, и одну треть — на центральный глаз. Таким образом, бехолдер, имеющий текущий потенциал 45 hp, может выдержать, прежде чем будет убит, 30 hp (удары, направленные на тело чудовища) и 15 hp, прежде чем центральный глаз ослепнет. При этом вероятность того, что атакующий попадет в тело чудовища, составляет 0.75, в центральный глаз — 0.1, в один из малых глаз — 0.05.

**Социальный статус.** Бехолдер — это отвратительное и агрессивное существо. Как правило, он атакует немедленно, без предупреждения. Но если бой предстоит с достаточно сильным противником, имеется приблизительно 50% за то, что монстр вступит в переговоры (либо о перемирии, либо об уплате монстром выкупа за то, что он не будет атакован). Конкретный вид договора зависит от атакующих возможностей Вашей стороны.

**Экология.** Малые глаза чудовища представляют собой важный ингредиент в алхимической практике. Они используются для изготовления *зелья левитации* и могут быть проданы магом или алхимиком по 50 золотых (golden pieces — gp) каждый.

### Глаз глубин

Это создание является, по-видимому, дальним предком бехолдера и встречается только в океанских впадинах.

Глаз глубин имеет две клешни, подобные крабьим, которые причиняют повреждения от 2 до 8 очков каждое, и зубастый рот, который может съесть от одного до шести очков.

Основным оружием этого чудовища также являются глаза. Большой центральный глаз монстра излучает конус слепящего света диаметром от 5 ярдов в начале до 30 ярдов в конце. Дальность действия луча составляет 20 ярдов. Попавший в зону действия луча должен быть защищен от действия яда или он будет парализован на 2-8 раундов схватки.

Глаз глубин имеет также два малых глаза, которые позволяют ему, действуя одновременно, создавать *иллюзию* (illusion). Если же малые глаза функционируют независимо, они способны выполнять функции *«очаровать гуманоида»* и *«очаровать чудовище»*.

Монстр имеет класс защиты 5, включая большой и малые глаза. Если в ходе битвы малые глаза повреждены, они отрастают по одному в неделю. ■

| | Бехолдер | Глаз глубин |
|---|---|---|
| **Климат/ местность** | Умеренный, влажный | Морские глубины |
| **Частота появления** | Очень редко | Очень редко |
| **Организация** | Одиночки | Одиночки |
| **Цикл активности** | Круглосуточно | День |
| **Пища** | Всеядный | Всеядный |
| **Разум** | Исключительный (15-16) | Выделяющийся (11-12) |
| **Сокровища** | I, S, T | R |
| **Мышление** | Силы мрака (lawful evil) | Силы мрака (lawful evil) |
| **Количество одновременно встречающихся** | 1 | 1 |
| **Класс защиты** | 0/2/7 | 5 |
| **Передвижение** | Полет 3 (B) | Плавание 6 |
| **Защитный потенциал** | 45-75 hp | 10-12 hp |
| **Количество атак за один раунд** | 1 | 1 |
| **Сила удара** | 2-8 | 2-8/2-8/1-6 |
| **Специальные атаки** | Магия | См. в тексте |
| **Специальная защита** | Луч защиты от магии | Нет |
| **Защита от магии** | Специальная | Нет |
| **Размер** | Средний (4-6 футов) | От малого до среднего (3-5 футов) |
| **Моральный статус** | Фанатик (18) | Чемпион (15) |
| **Опыт, приобретаемый при победе над монстром** | 14000 | 4000 |

НТО

Софт · Москва

*предлагает широкий спектр редакционно-издательских и полиграфических работ:*

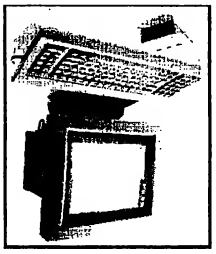
- литературное редактирование
- разработку рекламы для печатной продукции

*изготовление:*

- бланков, листовок
- брошюр, буклетов, проспектов
- комплектов документации для программных продуктов
- сертификатов ценных бумаг
- визиток, наклеек, этикеток, приглашений
- слайдов
- изданий за счет авторов

**Наш адрес:**
**117419 Москва, а/я 765.**
**Тел. (095) 238-99-33, 238-32-78.**

# Двенадцать программ с дублями и эпиграфами, или три триады программирования, а точнее третий лишний, а второй неправильный



*В. Очков*

Вот первые три дубля (на трех языках: *BASIC — C — Pascal*) программы с эпиграфом.

**Листинг 1. "Бог любит троицу"**

```
' Программа 1a (QBASIC)
DECLARE FUNCTION F (X)
INPUT "A, B, Погрешность"; A, B, Eps
Ya = F(A)                    ' Расчет Y на левом конце отрезка
DO WHILE B - A > Eps         ' Начало цикла "пока"
  X = (A + B) / 2            ' Делим отрезок пополам
  IF Ya * F(X) > 0 THEN      ' Альтернатива
       A = X                 ' Левое плечо альтернативы
       ELSE B = X            ' Правое плечо альтернативы
  END IF                     ' Конец альтернативы
LOOP                         ' Конец цикла "пока"
PRINT "Y = 0 при X = "; X    ' Ответ
END

FUNCTION F (X)
  F = X ^ 2 - 3
END FUNCTION
```

```
// Программа 1b (C)
#include <iostream.h>      // Подключение библиотеки потоков C++
double F (double);
void main(void) {          // Точка входа в программу
double A,B,Eps,YLeft,X;
cout << "Введите A, B, погрешность: "; cin >> A >> B >> Eps;
YLeft = F(A);              // Расчет Y на левом конце отрезка
while(B-A£ps) {            // Начало цикла "пока"
  X = (A+B)/2;             // Делим отрезок пополам
  if (YLeft*F(X) > 0)      // Альтернатива
       A = X;              // Левое плечо альтернативы
       else B = X;         // Правое плечо алтернативы
}                          // Конец цикла "пока"
cout << "Y = 0 при X = " << X;
}

double F(double X) {
return X*X-3;              //Анализируемое уравнение
}
```

```
{Программа 1c (Pascal)}
function F(X: real): real; begin
  F: = X*X-3;                {Анализируемое уравнение}
end; {end function}

var A, B, Ya, X, Eps: real;
begin
  write('A ? '); readln(A);
  write('B ? '); readln(B);
  write('Погрешность ? '); read(Eps);
  Y: = F(A);                 {Расчет Y на левом конце отрезка}
  while B - A > Eps do begin {Начало цикла "пока"}
       X: = (A + B) / 2;     {Делим отрезок пополам}
  if Ya * F(X) > 0 then      {Альтернатива}
       A: = X                {Левое плечо альтернативы}
       else B: = X;          {Правое плечо альтернативы}
                             {Конец альтернативы}
  end;                       {Конец цикла "пока"}
  writeln('Y = 0 при X = ', X);
end.
```

При поиске корня алгебраического уравнения методом половинного деления используется первая триада статьи — три основные алгоритмические конструкции: следование, повторение и выбор.

В эпиграфе программы 1 следовало бы написать не «Бог», а «Э. Дейкстра», имея в виду его три алгоритмические конструкции, упоминаемые в основной структурной теореме. И дело здесь не только в триаде (**следование — повторение — выбор**) [1]-[6], но и в том, что на эту теорему уповают уже более 30 лет, веря в нее, как в Господа Бога.

Вспомним ее.

«Алгоритм любой сложности можно реализовать, используя только три конструкции: **следование** (оператор за оператором), **повторение** (цикл) и **выбор** (альтернатива)».

Алгоритмических управляющих конструкций, позволяющих исключать метки из программ, конечно, намного больше трех. К трем главным следует добавить и неполную альтернативу, и вызов процедуры (особенно рекурсивной), и множественное ветвление, и работу с функциями пользователя. Тем не менее все они объявляются вспомогательными (первое следствие теоремы Э. Дейкстры), облегчающими и ускоряющими кодирование алгоритмов, но без которых при особом желании (как будет показано в этой статье) можно обойтись. Кроме того, из всех разновидностей циклов (цикл с предпроверкой, цикл с постпроверкой, цикл с параметром и другие) в качестве основного цикла был использован цикл «пока», а остальные были объявлены вспомогательными. Это, например, отразилось на языке *FRED* в составе интегрированного пакета *Framework* [7], где есть только цикл «пока»

@while(булево выражение, тело цикла)

и нет ни цикла «до», ни цикла с параметром.

**Листинг 2. "Старый конь борозды не испортит"**

```
10 REM Программа 2a (GW-BASIC)
20 INPUT "A, B, C"; A, B, C
30 D = B ^ 2 - 4 * A * C
40 IF D > = 0 THEN PRINT "X1 = "; (-B + SQR(D)) / 2 / A; " X2 = "; (-B -
SQR(D)) / 2 / A ELSE PRINT "Корней нет"
50 END
```

```
// Программа 2b (C)
#include <iostream.h>
#include <math.h>
void main(void)
double A,B,C,D;
cout << "A, B, C : "; cin >> A >> B >> C;
D = B*B-4*A*C;
if (D > = 0)
  cout << "X1 = " << (-B+sqrt(D))/2/A << "\n" \
       << "X2 = " << (-B-sqrt(D))/2/A;
  else cout << "Корней нет.\n";
}
```

```
{Программа 2c (Pascal)}
var A,B,C,D: real;
begin
  write('A:'); readln(A);
  write('B:'); readln(B);
  write('C:'); readln(C);
  D: = B*B-4*A*C;
  if D > = 0 then
  begin
       writeln('X1 = ', (-B+sqrt(D))/2/A);
       writeln('X2 = ', (-B-sqrt(D))/2/A);
  end
  else writeln('Корней нет');
end.
```

Поиск действительных корней квадратного уравнения — это старый классический пример, кочующий из одного учебника по программированию в другой и попавший в нашу статью для иллюстрации полной альтернативы.

**Листинг 3. "Любопытство не порок, а ... инструмент исследователя"**

```
' Программа 3a (QBASIC)
INPUT "A, B, C"; A, B, C
D = B ^ 2 - 4 * A * C
Flag = -1
DO WHILE D > = 0 AND Flag
  PRINT "X1 = "; (-B + SQR(D)) / 2 / A;
  PRINT " X2 = "; (-B - SQR(D)) / 2 / A
  Flag = 0
LOOP
DO WHILE Flag
  PRINT "Корней нет"
  Flag = 0
LOOP
END
```

```
// Программа 3b (C)
#include <iostream.h>
#include <math.h>
void main(void) {
enum BOOL {FALSE,TRUE};
double A,B,C,D;
enum BOOL Flag = TRUE;
cout << "A, B, C : "; cin > > A > > B > > C;
D = B*B-4*A*C;
while((D > = 0)&&Flag) {
  cout << "X1 = " << (-B+sqrt(D))/2/A << "\n" \
       << "X2 = " << (-B-sqrt(D))/2/A;
  Flag = FALSE;
}
while(Flag) {
  cout << "Корней нет.\n";
  Flag = FALSE;
}
}
```

```
{Программа 3c (Pascal)}
var A,B,C,D: real; Flag: boolean;
begin
  write('A:'); readln(A);
  write('B:'); readln(B);
  write('C:'); readln(C);
  D: = B*B-4*A*C;
  Flag: = TRUE;
  while (D > = 0) and Flag do begin
       writeln('X1 = ', (-B+sqrt(D))/2/A);
       writeln('X2 = ', (-B-sqrt(D))/2/A);
       Flag: = FALSE;
  end;
  while Flag do begin
       writeln('Корней нет');
       Flag: = FALSE;
  end;
end.
```

Алгоритм поиска действительных корней квадратного уравнения здесь выступает уже не в качестве «старого коня», а в качестве подопытного кролика, которого лишили альтернативы, дав взамен два цикла.

Программы, приведенные в листингах 2 и 3, доказывают, что при особом желании можно обойтись без альтернативы: в программах листинга 2 она есть, а в программах листинга 3 — нет. Программам в листинге 3 хватило цикла «пока» и переменной *Flag*, заставляющей операторы тела цикла работать либо раз, либо ни разу.

Полную альтернативу можно заменить на два цикла «пока» буквально в два счета [8]. **Делай раз:** меняем полную альтернативу на две неполные. Такой заменой рекомендуется [1] исключать некоторые метки в *Fortran* программах. **Делай два:** меняем каждую из двух неполных альтернатив на цикл «пока», тело которого выполняется либо раз, либо ни разу.

Итак, третий (альтернатива) — лишний. Теперь программами, приведенными в листингах 4-7, поясним, что значит **«а второй неправильный»** в названии статьи.

**Листинг 4. "Искусство – это чувство меры"**

```
10    REM Программа 4a (GW-BASIC)
20    DEF FNY (X) = X ^ 2 - 3
30    INPUT "A, B, Погрешность"; A, B, Eps: GOSUB 100: GOSUB 110
40    : IF ABS(B - A) < = Eps GOTO 80: REM Задача решена
50    IF Y1 < Y2 THEN B = X2: X2 = X1: Y2 = Y1: GOSUB 100: GOTO 70
60    A = X1: X1 = X2: Y1 = Y2: GOSUB 110
70    : GOTO 40: REM Новое приближение к минимуму
80    : PRINT "Y минимальный = "; FNY(X1); " при X = "; X1
90    STOP: REM Конец основной программы
100   : X1 = .618 * A + .382 * B: Y1 = FNY(X1): RETURN
110   : X2 = .382 * A + .618 * B: Y2 = FNY(X2): RETURN
120   END
```

```
// Программа 4b (C)
#include <iostream.h>
#include <math.h>
double Y(double);
void Calc1();
void Calc2();
double X1,X2,Y1,Y2,A,B; // глобальные переменные
void main(void) {
double Eps;
cout << "A, B, Погрешность :";cin > > A > > B > > Eps;
Calc1();Calc2();
Label_1: if(fabs(B-A) < = Eps) goto Label_3; // Задача решена
if (Y1 < Y2) {
  B = X2;X2 = X1;Y2 = Y1;
  Calc1();
  goto Label_2;
}
A = X1;X1 = X2;Y1 = Y2;
Calc2();
Label_2: goto Label_1; // Новое приближение к минимуму
Label_3: cout << "Y минимальный = " << Y(X1) << " при X = " << X1;
}

double Y(double X) {
return X*X-3;
}

void Calc1() {
X1 = 0.618*A+0.382*B;
Y1 = Y(X1);
}

void Calc2() {
X2 = 0.382*A+0.618*B;
Y2 = Y(X2);
}
```

Попросите знакомого художника разделить отрезок на две неравные части. Можете быть уверены — он это сделает в золотом соотношении. Подобное чувство меры позволяет компьютеру так делить интервал неопределенности *AB*, что при очередном приближении к минимуму высчитывается только одно значение анализируемой функции.

**Листинг 5. "Имя – это повесть из одного названия"**

```
' Программа 5a (QBASIC):
DEF FNY (X) = X ^ 2 - 3
INPUT "A, B, Погрешность"; A, B, Eps
GOSUB RIGHT                          ' "Правая" подпрограмма
GOSUB LEFT                           ' "Левая" подпрограмма
DO WHILE ABS(B - A) > Eps
  DO                                 ' Альтернатива
     IF Y1 < Y2 THEN B = X2: X2 = X1: Y2 = Y1: GOSUB RIGHT: EXIT DO
     A = X1: X1 = X2: Y1 = Y2: GOSUB LEFT: EXIT DO
  LOOP
LOOP                                 ' Конец цикла "пока"
PRINT "Y минимальный = "; FNY(X1); " при X = "; X1
STOP
RIGHT:                               ' "Правая" метка
  X1 = .618 * A + .382 * B: Y1 = FNY(X1)
RETURN
LEFT:                                ' "Левая" метка
  X2 = .382 * A + .618 * B: Y2 = FNY(X2)
RETURN
END
```

```
// Программа 5b (C)
#include <iostream.h>
#include <math.h>
double Y(double);
void Right();
void Left();
double X1,X2,Y1,Y2,A,B;
void main(void) {
double Eps;
cout << "Введите A, B, погрешность: "; cin >> A >> B >> Eps;
Right();                             // "Правая" подпрограмма
Left();                              // "Левая" подпрограмма
while(fabs(B - A) > Eps) {
  for(;;) {                          // Альтернатива
      if (Y1 <Y2) {
          B = X2;X2 = X1;Y2 = Y1;
          Right();
          break;
      }
      A = X1;X1 = X2;Y1 = Y2;
      Left();
      break;
  }
}                                    // Конец цикла "пока"
cout << "Y минимальный = " << Y(X1) << " при X = " << X1 << "\n";
}

double Y(double X) {
  return X*X-3;
}

void Right() {                       // "Правая" метка
  X1 = 0.618*A+0.382*B;
  Y1 = Y(X1);
}

void Left() {                        // "Левая" метка
  X2 = 0.382*A+0.618*B;
  Y2 = Y(X2);
}
```

Дав метке (переменной, функции, процедуре) «звучащее» имя (*LEFT* — левый, *RIGHT* — правый), никогда не забудешь, «какая повесть ею озаглавлена», — что делается в соответствующих подпрограммах, где сужается левый (*A-X1*) или правый (*X2-B*) интервал неопределенности.

Исключить альтернативу можно и по-другому — использованием не цикла «пока», а цикла с выходом из середины. Это проиллюстрировано программами в листингах 4 и 5, осуществляющими поиск минимума одномерной унимодальной функции методом золотого сечения.

В программе 5a был использован не просто цикл, а цикл с несколькими выходами из середины и со «словами прощания». Его следует считать, с одной стороны, «главным» циклом, так как он легко трансформируется и в цикл «пока», и в цикл «до», с другой — гибридом цикла и альтернативы. Во втором случае цикл становится одноразовым — для страховки в программе 5a лучше написать не *LOOP*, а *LOOP UNTIL 1=1*.

Выведенную нами новую «породу» управляющих конструкций алгоритмов с языка *QBASIC* на языки *C* и *Pascal* так просто не переведешь: «Можно быть большим католиком, чем папа римский, и более структурированным языком, чем *C* и *Pascal*». На языках *C* и *Pascal* цикл с выходом из середины можно получить только через его «запихивание» в прокрустово ложе либо цикла *for* (программа 5b), либо цикла *while* (*while 2*2=4 ...*), либо цикла *do* (*repeat ... until 1=2*). Цикл же с несколькими (двумя) выходами из середины позволяет исключить полную альтернативу (листинг 5) без дополнительной переменной *Flag* (листинг 3). Так что цикл «пока» занял второе место в основной структурной теореме неправомерно, а альтернатива (третий), как мы уже показали, совсем лишняя.

Весь фокус здесь в том, что альтернатива — это средство ускоренного «путешествия» по алгоритму только в одну сторону — сверху вниз и слева направо, если смотреть на листинг, а цикл — в обе. Отсюда и ненужность альтернативы (в теоретическом плане, конечно, а не в практическом).

Но ценность, а точнее, незаменимость, цикла с выходом из середины, конечно, не в том, что за счет его можно из программы исключить полную альтернативу без ввода новых переменных (сравните листинги 4 и 5), а в том, что выходов из цикла может быть несколько. Этого не избежать, когда значение управляющего логического выражения меняется вне цикла и есть опасение пропустить в нужный момент его изменение. Такое случается, если в программе задействованы элементы параллельного программирования, когда, например, управляющее циклом логическое выражение опирается на показание встроенных часов компьютера или на опрос состояния периферии (нажата ли какая-нибудь клавиша клавиатуры, включен ли принтер, вставлена ли в него бумага, есть ли в дисководе диск, отформатирован ли он и так далее).

Цикл с выходом из середины позволяет более просто убирать из программы метки без использования второй классической триады программирования — трех методов структурирования алгоритмов [1]: дублирования, ввода признака и разбега (листинги 6–9).

**Листинг 6. "Не откладывай на завтра то, что можно сделать сегодня"**

```
10    REM Программа 6a (GW-BASIC)
20    RANDOMIZE TIMER-32767
30    PRINT "Угадай число от 1 до 1000 за минимум попыток."
40    A = 1 + INT(1000*RND(1))
50    I = 0:REM Обнуление счетчика попыток
60:   I = I + 1
70    PRINT I;"-я попытка";
80    INPUT B
90    IF B > A THEN PRINT "Перелет":GOTO 60
100   IF B < A THEN PRINT "Недолет":GOTO 60
110   PRINT "Угадали"
120   END
```

```
// Программа 6b (C)
#include <stdlib.h>
#include <time.h>
#include <iostream.h>
inline double RND() {                // Функция, аналогичная оператору RND(1)
                                     // языка BASIC
return ((double)rand()/(double)RAND_MAX);
}
void main(void) {
int A,B,I = 0;
srand((unsigned)time(NULL)); // Инициализация генератора
                             // случайных чисел
RND();                       // "холостой ход"
cout << "Угадай число от 1 до 1000 за минимум попыток.\n";
A = 1 + (int)(1000*RND());
label_1: I++;
cout << I << "-я попытка\n";
cin >> B;
```

**Листинг 6 (окончание)**

```
if (B>A) {
  cout < < "Перелет\n";
  goto label_1;
}
if (B<A) {
  cout < < "Недолет\n";
  goto label_1;
}
cout < < "Угадали\n";
}
```

Подчеркнуть «сегодня» (в момент написания программы) вторую роль разметки *GW-BASIC* программ можно знаками двоеточия за номерами строк, которые «притягивают» к себе операторы *GOTO*, *GOSUB* и другие (см. также программы 4а, 8а и 10). Это поможет «завтра» при необходимости быстро избавиться от этих номеров при объединении строк или при переходе к работе с языком *QBASIC*, *C* или *Pascal*.

**Листинг 7. "Голь на выдумки хитра"**

```
' Программа 7а (QBASIC)
RANDOMIZE TIMER - 32767
PRINT "Угадай число от 1 до 1000"
A = 1 + INT(1000 * RND(1))
I = 0 ' Обнуление счетчика попыток
DO ' Заголовок цикла "до"
  I = I + 1
  PRINT I; "-я попытка";
  INPUT B
  IF B > A THEN PRINT "Перелет"  ' IF A = B THEN EXIT DO
  IF B < A THEN PRINT "Недолет"  ' IF B > A THEN PRINT "Перелет"
  ELSE PRINT "Недолет"
LOOP UNTIL A = B                 ' LOOP
PRINT "Угадали"
END
```

```
// Программа 7b
#include <stdlib.h>
#include <time.h>
#include <iostream.h>
                                 // enum {FALSE,TRUE};
inline double RND() {
return ((double)rand()/(double)RAND_MAX);
}

void main(void) {
int A,B,I = 0;

 srand((unsigned)time(NULL));RND();
cout < < "Угадай число от 1 до 1000 за минимум попыток.\n";
A = 1+(int)(1000*RND());
do { // Заголовок цикла "до"
  I++;
  cout < < I < < "-я попытка \n";
  cin > > B;
  if (B>A) cout < < "Перелет\n";  // if (A= =B) break;
  if (B<A) cout < < "Недолет\n";  // if (B>A) cout < < "Перелет\n";
                                  // else cout < < "Недолет\n";
}
while (A! = B);                   // while (TRUE);
cout < < "Угадали\n";
}
```

Отсутствие в наборе встроенных функций языков *BASIC* и *C* генератора истинно случайных чисел заставляет «хитрых» программистов использовать встроенные часы компьютера (*TIMER*) и задавать тем самым истинно случайное значение переменной *A*.

Программы 6 и 7 позволяют сыграть с ЭВМ в игру «Угадай число», когда машина загадывает целое число в диапазоне от 1 до 1000 (строка 40 программы 6а), которое человек должен отгадать за минимум попыток. Этой игрой часто иллюстрируют принцип бинарного (двоичного) поиска. Придерживаясь оптимальной стратегии, человек должен делить интервал неопределенности пополам, а число, оказавшееся в середине, сообщать машине в качестве очередной догадки (строка 80 программы 6а). По ответам ЭВМ «Перелет» или «Недолет» человек уменьшает интервал неопределенности вдвое. В *GW-BASIC* программу 6а заложена естественная последовательность действий: если догадка человека больше (строка 90) или меньше (строка 100) задуманного числа, то (*THEN*) машина сообщает об этом и сразу оператором безусловного перехода возвращается к запросу новой догадки. Программы в листинге 6 можно отструктурировать, продублировав некоторые действия и тем самым преобразовав в программы листинга 7. Здесь операторы безусловного перехода к меткам убраны, но появился цикл «до» с лишней, на первый взгляд, функцией. Ведь если значение *B* не больше значения *A* и значение *B* не меньше значения *A*, то незачем дополнительно проверять равенство чисел. Но эта дублирующая операция структурировала программу: теперь алгоритм игры реализуется через цикл «до», тело которого содержит две последовательные неполные (с одним плечом) альтернативы. Комментарии программ в листинге 7 показывают, что замена цикла «до» на цикл с выходом из середины намного облегчает и упрощает структурирование алгоритма, восстанавливая у него естественную последовательность действий и исключая дублирование.

Цикл «до» считается вспомогательной управляющей конструкцией и используется тогда, когда булево выражение, управляющее циклом, приобретает смысл только после первого выполнения его тела, как в программах в листинге 7. Заменяется цикл «до» на цикл «пока», если в этом есть необходимость, как при программировании на языке *FRED* (см. ранее), искусственным заданием логическому выражению нужного значения: $B = A + 1$, например, если иметь в виду программы листинга 6. Это похоже на ситуацию «вне игры» (*offside*) в футболе, когда защитники пробегают вперед и оставляют нападающего одного у ворот.

Второй метод структурирования (ввод признака) иллюстрируется программой, решающей известную задачу английского физика Поля Дирака (1902-1984).

«Три рыбака легли спать, не поделив улова. Проснувшийся ночью первый рыбак решил уйти, взяв свою долю. Но число рыб не делилось на три. Тогда он выбросил одну, а из остатка забрал треть. Второй и третий рыбаки поступили аналогичным образом. Спрашивается, какое наименьшее количество рыб соответствует условию задачи».

Поль Дирак был мастер давать разным существительным приставку «анти» — античастица, например. И в задаче о рыбаках и рыбке он, по-видимому, не изменил своей привычке, оригинально решив ее: минус две рыбы. Выбрасываем одну антирыбу — получаем минус три, забираем одну (треть улова) — получаем минус две и так далее. Но в ответ вкралась ошибка, которую можно объяснить только тем, что у Дирака не было компьютера.

**Листинг 8. "Красиво жить не запретишь"**

```
10    REM Программа 8а (GW-BASIC)
20    INPUT "Число рыбаков";Fishers
30    INPUT "Дельта рыб";DeltaFish
40    INPUT "Начальное число рыб";StartFish
50:   P = StartFish:REM Первый рыбак насчитал StartFish рыб
60    FOR I = 1 TO Fishers:REM Перебор всех рыбаков
```

**Листинг 8 (окончание)**

```
70        P = P + DeltaFish:REM Каждый выбросил или подловил DeltaFish рыб
80        P = P-P/Fishers:REM и взял свою долю
90        IF P > INT(P) THEN StartFish = StartFish-1:GOTO 50
100   NEXT I:REM Просыпается следующий рыбак
110   PRINT"Ответ "StartFish" рыб
120   END
```

```
// Программа 8b (C)
#include <iostream.h>
void main(void) {
int Fichers,DeltaFish,StartFish,I;
double P;
cout < < "Число рыбаков :"; cin > > Fichers;
cout < < "Дельта рыб :"; cin > > DeltaFish;
cout < < "Начальное число рыб :"; cin > > StartFish;
Label_1: P = StartFish;        // 1-й рыбак насчитал StartFish рыб
for (I = 1;I < = Fichers;I + +) {   // Перебор всех рыбаков
  P + = DeltaFish;             // Каждый выбросил или подловил DeltaFish рыб
  P- = P/Fichers;              // и взял свою долю
  if (P! = (int)P) { StartFish = StartFish-1; goto Label_1;}
}                              // Просыпается следующий рыбак
cout < < "Ответ " < < StartFish < < " рыб\n";
}
```

Даже в неструктурированной *GW-BASIC* программе отступами от номеров строк можно попытаться «красиво» выделить структурные управляющие конструкции алгоритма.

**Листинг 9. "Timeo Danaos et dona ferentes"**

```
' Программа 9a (QBASIC)
INPUT "Число рыбаков"; Fishers
INPUT "Дельта рыб "; DeltaFish
INPUT "Начальное число рыб"; StartFish
StartFish = StartFish + 1                    ' Ничего нет
DO
  StartFish = StartFish - 1                  ' Ничего нет
  Flag = -1                                  ' Ничего нет
  P = StartFish
  FOR I = 1 TO Fishers
      P = P + DeltaFish
      P = P - P / Fichers                    ' Здесь ошибка
      IF P > INT(P) THEN Flag = 0            ' IF P > INT(P) THEN EXIT FOR
  NEXT
                                             ' IF P = INT(P) THEN EXIT DO
                                             ' StartFish = StartFish - 1
LOOP UNTIL Flag                              ' LOOP
PRINT "Ответ "; StartFish; "рыб"
END
```

```
// Программа 9b (C)
#include <iostream.h>
void main(void) {
enum BOOL {FALSE,TRUE};
int Fishers,StartFish,DeltaFish,I;
double P;
enum BOOL Flag;
cout < < "Число рыбаков :";cin > > Fishers;
cout < < "Дельта рыб : "; cin > > DeltaFish;
cout < < "Начальное число рыб :"; cin > > StartFish;
StartFish + = 1;
do {
  StartFish- = 1;
  Flag = FALSE;
  P = StartFish;
  for(I = 1;I < = Fishers;I + +) {
      P + = DeltaFish;
      P- = P/Fishers;
      if (P! = (int)P) Flag = TRUE;   // if (P > (int)P) break;
  }
                                      // if (P = = (int)P) break;
                                      // StartFish- = 1
}
while(Flag);                          // while(TRUE);
cout < < "Ответ " < < StartFish < < " рыб\n";
}
```

«Бойтесь данайцев, дары приносящих»: языки семейства *BASIC* не требуют объявления переменных. Такой «подарок» часто бывает причиной очень неприятной ошибки — появления в программе «троянского коня» — переменной-мутанта *Fichers* вместо правильного *Fishers*, например.

Программы в листингах 8 и 9 решают задачу о рыбаках и рыбке методом перебора, когда задается начальное число рыб (строка 40 программы 8а) и проверяется, удовлетворяет ли оно условиям честного дележа (цикл с параметром в строках 60-100 программы 8а). Если да, то ответ готов (строка 110), нет — начальное число рыб уменьшается на единицу (строка 90), а расчет повторяется. Рано или поздно машина ответ найдет. При этом ее, как и Поля Дирака, не будет смущать тот факт, что число рыб стало величиной отрицательной. Нормального человека такой нюанс смутил бы, конечно, но нормальный человек и до античастиц не додумался бы: талант — это аномалия.

В *GW-BASIC* программе в листинге 8а досрочное прерывание цикла с параметром «разваливает» структуру. Спасти ее можно, введя в программу **переменную-признак** (*Flag* — листинг 9). Перед входом в цикл «флаг» поднимается: «презумпция невиновности». Если же хоть раз остаток от дележа улова будет нецелым числом, то «флаг» опускается. Это будет сигналом к повторению счета с уменьшенным на единицу числом рыб. Алгоритм машинного решения задачи Дирака стал структурированным: в цикл «до» вложен цикл с параметром и с неполной альтернативой.

В программах листинга 9 использован и третий метод структурирования — **разбег** (шаг вперед — шаг назад): начальное число рыб сначала увеличивается на единицу, а потом сразу, но уже в цикле, на эту же единицу уменьшается. Такой ход в программе подобен поведению прыгуна на разбеге: шаг назад, а потом разбег. Удаление или замена в программах листинга 9 некоторых операторов на те, которые записаны в комментариях, позволяет, так же как и в программах листинга 7, структурировать программу без использования триады классических методов.

Расчеты по программам в листингах 8 и 9 дают не одно, а целый ряд решений задачи:

..., -110, -83, -56, -29, -2, 25, 52, ...

Никто не будет спорить, что минус 29, а тем более минус 110 меньше, чем минус 2. Да, жаль, что у Дирака не было компьютера.

На языках *C* и *Pascal* цикла с выходом из середины нет, и это печально. Но при необходимости он реализуется двумя способами. Во-первых, прервать цикл в любом месте можно с помощью метки и оператора условного перехода (*if ... goto ...*). Те же, на кого метка в программе действует, как красная тряпка на быка, поступают, как говорил Даниил Хармс, «самым остроумным способом», а именно: тело цикла записывается процедурой с вкраплениями *if ... then exit*, то есть с командами «если ..., то прерви выполнение процедуры и вернись в основную программу или в процедуру уровнем выше к оператору, находящемуся за именем процедуры». Многие программисты к основным структурным управляющим конструкциям относят и вызов процедуры. И не рекурсивный (нет такой рекурсии, какую нельзя было бы свести к итерации), а простой вызов процедуры, к которому прибегают в описанном случае. На языке *QBASIC* цикл можно прервать в любом месте и не в одном (*EXIT DO* или *EXIT FOR*). Так же бесцеремонно *QBASIC* позволяет поступать и с процедурами (*EXIT SUB*), и с функциями (*EXIT FUNCTION*). Язык *C* (а вслед за ним и язык *Turbo Pascal 7.0*) проявляет еще большую

гибкость: в нем цикл можно прервать, передав управление программой либо в конец цикла (*if ... break*), либо в его начало (*if ... continue*).

Если читатель еще не совсем запутался, то вот какую «ясность» в проблему минимального числа управляющих конструкций алгоритмов (суть основной структурной теоремы Дейкстры) внесла переведенная с английского и выпущенная в 1989 г. издательством «Мир» книга «Языки программирования Ада, Си и Паскаль. Сравнение и оценка». В ней на страницах 72 и 73 сказано: «В соответствии с генеалогией управляющих структур <...> циклы с использованием оператора завершения *Break* и оператора продолжения *Continue* относятся к классу $DREC^1$. (Такую аббревиатуру европеец не мог придумать: *a dreg* по-английски «отбросы»; *die Dreck* по-немецки «грязь», «дерьмо» — примечание ред.). Теоремы, доказанные Р. Косараю (точно, не европеец — японец), показывают, что управляющие структуры, относящиеся к классу $DREC^1$, не могут быть эмулированы с помощью управляющих структур, относящихся к классу *D* (название этого класса образовано от первой буквы фамилии Э. Дейкстры — автора основной структурной теоремы), которые составляются из произвольного числа условных операторов *if*, операторов цикла *while* и их конкатенаций. На самом деле управляющие структуры, относящиеся к классу $DREC^i$ (имеющие в своем составе оператор выхода *exit(i)*, обеспечивающий выход на *i* уровней вверх, и оператор *cycle(i)*, обеспечивающий выполнение цикла на *i*-м уровне вложенности, как, например, в языке Блисс), являются более мощными, чем управляющие структуры, относящиеся к классу $DREC^{i-1}$.<...> Однако программы, содержащие циклы с использованием операторов *exit* и *cycle*, оказываются намного сложнее для понимания. Анализ показывает, что необходимость использования оператора *exit* возникает крайне редко. Необходимость наличия управляющих структур более высокого уровня, чем управляющие структуры класса *D*, до сих пор не доказана». Вот так-то. Начали «за упокой» основной структурной теоремы, а кончили «за здравие».

**Листинг 10. "Жадность фраера погубит"**

```
10    REM Программа 10 (GW-BASIC)
20    DEF FNTax (a) : REM Заголовок функции Tax
30        IF a < 42000 THEN T = .12 * a: GOTO 150
40        IF a < 84000 THEN T = 5040 + .15 * (a - 42000): GOTO 150
50        REM "А то что ж: один в семи комнатах расселился,
55        REM штанов у него 40 пар,
60        REM а другой шляется, в сорных ящиках питание ищет", —
70        REM ремарка булгаковского Полиграфа Полиграфовича
75        REM Шарикова из "Собачьего сердца".
80        IF a < 120000 THEN T = 11340 + .2 * (a - 84000): GOTO 150
90        REM Наши парламентарии спутали заработанную плату
95        REM со сверхприбылью монополии.
100       IF a < 180000 THEN T = 18540 + .3 * (a - 120000): GOTO 150
110       REM Комментарий, не пропущенный в статью
115       REM внутренним цензором автора.
120       IF a < 300000 THEN T = 36540 + .4 * (a - 180000): GOTO 150
130       IF a < 420000 THEN T = 84540 + .5 * (a - 300000): GOTO 150
140       T = 144540 + .6 * (a - 420000)
150:  REM Конец конструкции "Выбор"
160       FNTax = T: REM Присвоение найденной величины функции
180   END DEF: REM Конец функции Tax
190   INPUT " Чистые (рубли в год)"; Ch:REM Запрос суммы "чистых"
200   Tmp1 = Ch: REM Дублирование Ch во временной переменной
210:  REM Начало цикла с выходом из середины
220       Tmp2 = Ch + FNTax(Tmp1): REM Приближение к корню
230       IF ABS(Tmp1 - Tmp2) < .001 GOTO 260
240       Tmp1 = Tmp2: REM Подготовка к новому приближению
250   GOTO 210: REM Конец цикла с выходом из середины
260:  adjPay = Tmp2: REM Присвоение найденной величины функции
270   PRINT USING " Грязные = #########.## руб"; adjPay
280   PRINT USING " Налог = ########.## руб "; adjPay - Ch
290   PRINT USING " Налог = ##.## %"; 100 * (adjPay - Ch) / adjPay
300   END
```

Программа 10 решает обратную налоговую задачу — по сумме «чистых» денег методом простых итераций рассчитывает сумму «грязных». Пример в оправдание эпиграфа: чтобы в год заработать «чистый» миллион, нужно отдать государству 1,231,350 рублей, то есть 55.18 % от суммы «грязных».

**Листинг 11. "Богатые тоже плачут"**

```
' Программа 11 (QBASIC)
DECLARE FUNCTION TaxMarried (adjPay)
DECLARE FUNCTION TaxSingle (adjPay)
  INPUT "Грязные ($ в неделю) "; adjPay
  PRINT "Налог с холостого (незамужней) = "; TaxSingle(adjPay); "$"
  PRINT "Налог с женатого (замужней) = "; TaxMarried(adjPay); "$"
END

FUNCTION TaxMarried (adjPay)
'Расчет федерального налога с женатого (замужней)
a = adjPay
DO ' Заголовок конструкции "Выбор"
  IF a < 62 THEN T = 0: EXIT DO
  IF a < 657 THEN T = .15 * (a - 62): EXIT DO
  IF a < 1501 THEN T = 89.25 + .28 * (a - 657): EXIT DO
  IF a < 3695 THEN T = 325.57 + .33 * (a - 1501): EXIT DO
  T = 1049.59 + .28 * (a - 3695): EXIT DO
LOOP                                  ' Конец конструкции "Выбор"
TaxMarried = T
END FUNCTION

FUNCTION TaxSingle (adjPay)
'Расчет федерального налога с холостого (незамужней)
a = adjPay
DO ' Заголовок конструкции "Выбор"
  IF a < 21 THEN T = 0: EXIT DO
  IF a < 378 THEN T = .15 * (a - 21): EXIT DO
  IF a < 885 THEN T = 53.55 + .28 * (a - 378): EXIT DO
  IF a < 2028 THEN T = 195.51 + .33 * (a - 885): EXIT DO
  T = 572.7 + .28 * (a - 2028): EXIT DO
LOOP                                  ' Конец конструкции "Выбор"
TaxSingle = T
END FUNCTION
```

Программа 11 решает прямую налоговую задачу — по сумме «грязных» денег вызовом функции рассчитывает сумму «чистых». Отечественный «миллионер» (см. подпись к программе 10) в США зарабатывает 100 долларов в неделю и платит федеральный налог 11 долларов 85 центов, если он холостой, и 5 долларов 70 центов, если он женатый.

Еще одну структурную управляющую конструкцию алгоритмов **«Выбор»** иллюстрируют *BASIC* программы 10 и 11, решающие задачу о подоходном налоге. Под термином «грязные» понимается количество заработанных денег за вычетом не облагаемой налогом суммы. Конструкцию выбора (*ON ... GOTO ...* языка *GW-BASIC* и *SELECT ... END SELECT* языка *QBASIC*) заменяет серия альтернатив (*IF ... THEN ... GOTO ...*) (листинг 10), которую, в свою очередь, можно заменить на цикл со многими выходами из середины (программа 11).

Сравнивая программы 10 и 11, можно проследить не только за эволюцией структурных управляющих конструкции, но и за более интересными вещами, например, за изменением технологии программирования на языке *BASIC* [9]. Непрофессиональный программист, прибегающий к *GW-BASIC* для решения своей прикладной задачи (а профессиональные программисты *GW-BASIC* и за язык-то не считают), часто поступает подобно хоккеисту, вбрасывающему, недолго думая, шайбу в чужую зону в надежде на то, что потом в свалке удастся взять ворота противника. Так, например, в программе 10 программист первым оператором *INPUT «Чистые (рубли в год)»; Ch* «забросил» переменную *Ch* в ОЗУ машины, а затем «ринулся» за ней следом в надежде на то, что как-нибудь в «свалке» новых вводимых переменных (*Tmp1*, *Tmp2*

и других) и в коротких «передачах» *GOTO* и *GOSUB* удастся «забросить шайбу» — решить поставленную задачу. Среда *QBASIC*, не отвергая описанной тактики, предоставляет программисту и другую (программа 11). Можно «в своей зоне» не спеша разыграть комбинацию — написать и отладить процедуры-функции: *TaxMarried (adjPay)* и *TaxSingle (adjPay)*, а затем коротким броском — *PRINT TaxSingle(adjPay)* и *PRINT TaxMarried(adjPay)* — «отправить шайбу в ворота противника».

**Листинг 12. "Пока рак на горе не свистнет"**

```
' Программа 12 (QBASIC)
DECLARE SUB Bell (t$)
CLS
DO
  LOCATE 10, 30: PRINT TIME$
  IF MID$(TIME$, 4, 2) = "00" THEN IF RIGHT$(TIME$, 2) = "00" THEN
      Bell (TIME$)
LOOP                          ' LOOP UNTIL 1 = 2

SUB Bell (t$)
  h = VAL(LEFT$(t$, 2))
  IF h > 12 THEN h = h - 12
  IF h = 0 THEN h = 12
  FOR i = 1 TO h              ' i = 1: DO: IF i > h THEN EXIT DO
  PLAY "D P16"                ' Бой часов
  NEXT                        ' i = i + 1: LOOP
END SUB
```

Часы с боем имеют вечный завод, так как в программе задействован бесконечный цикл, прерывающийся, конечно, не по «свистку», а аккордом *CTRL-BREAK*.

Многие, решая задачу о рыбаках и рыбке (листинги 8 и 9), не получали ни дираковского (-2), ни нашего компьютерного (-29) ответов из-за того, что под числом интуитивно понимается что-то положительное и не меньше единицы. Такие «шоры» иногда мешают и гибкому использованию цикла с выходом из середины: подразумевается, что выходов из цикла может быть один или более. В программе 12 цикл имеет ноль выходов. Эта простейшая программа окажется трудным орешком для *Pascal* программистов, избегающих меток. Организация процедуры здесь не поможет, а выручит только «свисток рака» — конструкция *REPEAT ... UNTIL 1=2* (ремарки в листинге 12 показывают, как циклом «до» заменяется бесконечный цикл, а циклом с выходом из середины — цикл с параметром).

Более того, цикл с выходом из середины, если еще раз обратиться к дираковскому решению задачи о рыбаке и рыбке, может иметь и ... отрицательное число выходов, если под ними иметь в виду входы в цикл, минуя его заголовок. При этом можно получить новые структурные управляющие конструкции, еще более повышающие гибкость языка программирования. В свое время программисты отказались и от меток, и от допустимости вхождения в цикл, минуя заголовок. Потом метки были прощены, и им было позволено вернуться в программы. Не пришла ли пора реабилитации входов в цикл, минуя заголовки?

*P.S.*

*Магия цифры три не обошла стороной и автора, который в статье с тремя названиями поместил 12 (четырежды* **три***) программ, ссылается на 9 (трижды* **три***) литературных источников, упоминает шесть (дважды* **три***) языков программирования (***BASIC, Pascal, C, Fortran, FRED** *и таинственный* **Блисс***), а в выводах оставил ровно три (одиножды* **три***) пункта:*

*1. Основная структурная теорема в течение почти 30 лет давалась в неверной формулировке. Правильная редакция такая: «Алгоритм любой сложности можно реализовать, используя только две конструкции: следование и повторение».*

*2. Альтернативу следует относить к вспомогательным структурным управляющим конструкциям: ее можно заменить на цикл с двумя выходами из середины — с одним условным и со вторым безусловным.*

*3. Языки* **C** *и* **Pascal** *(включая и версию* **Turbo Pascal 7.0***) базируются на неполном наборе структурных управляющих конструкций, что влечет за собой в некоторых случаях немотивированное использование меток.*

*Сейчас программисты «больны новой болезнью» — объектно-ориентированным программированием с новой (третьей)* **триадой**: **полиморфизм, инкапсуляция** *и* **наследование**. *Все ли там логично и нельзя ли обойтись диадой — это тема отдельной статьи.* ■


Литература

1. Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ: Пер. с англ. – М.: Мир, 1985.
2. Вирт Н. Алгоритмы + структуры данных = программы: Пер. с англ. – М.: Мир, 1985.
3. Вирт Н. Систематическое программирование: Пер. с англ. – М.: Мир, 1977.
4. Дал У., Дейкстра Э., Хоор К. Структурное программирование: Пер. с англ. – М.: Мир, 1975.
5. Йодан Э. Структурное проектирование и конструирование программ: Пер. с англ. – М.: Мир, 1979.
6. Майерс Г. Отладка и тестирование программ: Пер. с англ. – М.: Мир, 1979.
7. Очков В.Ф., Пухначев Ю.В. Уроки для пользователей IBM PC. – М.: Финансы и статистика, 1992.
8. Очков В.Ф., Пухначев Ю.В. 128 советов начинающему программисту. – М.: Энергоатомиздат, 1992.
9. Очков В.Ф. Языки программирования GW-BASIC и QBASIC: сравнительное описание. – М.: Энергоатомиздат, 1992.

*Русскоязычная версия!*

SYMANTEC

*Диалог-МИФИ*
*095-320-3466*

*Merisel-Russia*
*095-276-9008*

*Перспективные технологии*
*095-256-6271*

*Трио Плюс*
*095-971-1204*

***Прекрасный текстовый процессор...***
***Для Windows™...***
***Для тех, кто хочет просто и красиво работать с текстом...***

Фирма ИНСАЙТ INSIGHT Comp.

Дистрибьютер **CLARION SOFTWARE** Corp.

# CLARION TopSpeed

Версии 2.1 и 3.0 Pascal, Modula-2, C, C++

*Идеальные средства создания заказного программного обеспечения.*

Семинары CLARION User Group;
для официальных пользователей -
"горячая линия"; учебные курсы.

*Предлагаем программное обеспечение фирм*
**Borland, Symantec, Microsoft, Corel, WordPerfect**

103489, Москва, Зеленоград, корп. 602а
тел. 536-69-00 факс 534-93-46
E-Mail: puh@insight.pccentre.msk.su | 2:5020/11
CompuServe: 100140, 1224

# ИНТЕРПРОКОМ

**дистрибутор фирм**
**NOVELL, LOTUS, GUPTA И MICRODYNE**
**предлагает**

NOVELL

сетевые операционные системы Novell NetWare,
DR DOS и UnixWare

Lotus

средства организации локальных и глобальных
систем электронной почты cc:Mail

Gupta

сетевые СУБД и средства разработки SQL приложений

Microdyne

сетевое оборудование и коммуникационные средства

Приглашаем к сотрудничеству по распространению
программных продуктов фирм Novell, Lotus и Gupta
на основе дилерских договоров

Адрес : 117036, г. Москва, ул. Дм. Ульянова 26/2
Тел.: 129-80-09, Телетайп: 207940 POKOM
Факс: 310-70-91; Телекс: 412249 INPRO SU

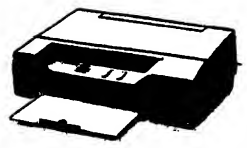
# ПЕРСПЕКТИВНЫЕ ТЕХНОЛОГИИ

## КАЧЕСТВО, НА КОТОРОЕ МОЖНО ПОЛОЖИТЬСЯ

# EPSON®

## ПРИНТЕРЫ

### МАТРИЧНЫЕ



LX-100 ( 9-pin, 80 col, автоподача бумаги)
LQ-100 (24-pin, 80 col, автоподача бумаги)

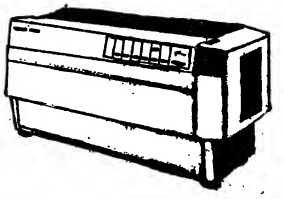
FX-11070 ( 9-pin, 80 col, автоподача бумаги);
LQ-1070 (24-pin, 80 col, автоподача бумаги);

**Идеальное решение для офиса.**

**Лучшая модель для офиса.**

**Удивительное сочетание скорости, качества и красоты.**

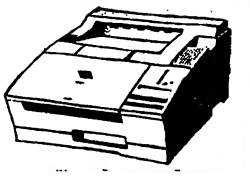
### БАНКОВСКИЙ



DFX-8000 матричный, (18-pin) высокоскоростной банковский принтер, скорость печати 1066 з/с, два варианта подачи бумаги, печать до пяти копий на самокопирующейся бумаге.

**Для очень серьезных пользователей**

### ЛАЗЕРНЫЕ



**Недорогие расходные материалы.**

EPL-5000 (emul, HPIIP, на 6000 копий)
EPL-5200 (emul, HPIIIP, на 6000 копий)

## PARADOX FOR WINDOWS

*Мощная реализация реляционной модели*

**Paradox For Windows обеспечивает логическую целостность хранимых в базе данных, а также позволяет держать их на диске в зашифрованном виде. Новые типы полей - Мето, графический, OLE, BLOB позволяют применять Paradox For Windows в областях, непривычных для реляционных СУБД.**

*Полное использование графики MS Windows*

**как и среда Paradox For Windows, так и приложения, создаваемые с его помощью, используют все средства конструирования графических пользовательских интерфейсов.**

*Простота в обращении:*

**Графический "запрос по образцу". Визуализация данных во всех режимах работ при проектировании схемы базы данных (Linking Export); при создании форм, запросов и отчетов (Form, Report & Query Exports); изображении структуры объектов программы (Object Tree). Все это позволяет создавать структурно сложные базы данных и большие программные комплексы.**

**Драйвер русификации поставляется бесплатно.**
**СКИДКИ ДЛЯ ДИЛЕРОВ 35-40%.**

Москва, Звенигородское шоссе, д. 9 (м. "Улица 1905 г."). Факс: (095) 259-27-27. Тел.: (095) 256-42-42, 256-40-30.

# Прислушайтесь к мнению ведущих мировых авторитетов о компьютерах фирмы

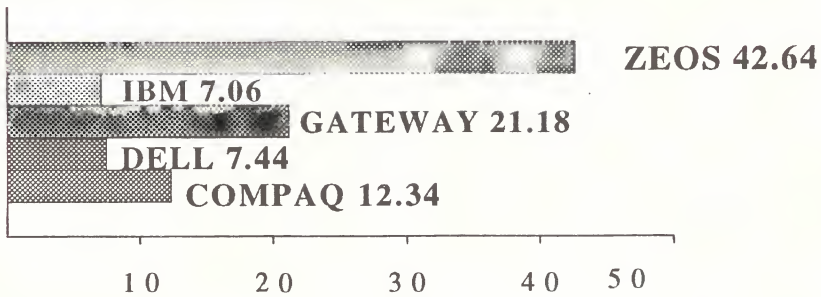
# ZEOS®

## INTERNATIONAL, LTD.

## *Вот самые последние оценки*

**ИТОГИ BENCHMARK-ТЕСТИРОВАНИЯ "GRAPHICS WINMARK" ЛАБОРАТОРИИ ЖУРНАЛА PC MAGAZINE, 1993 Г.**

(МЕГАПИКСЕЛЕЙ В СЕКУНДУ)




PC MAGAZINE EDITORS' CHOICE

*"Zeos... просто смел всех своих конкурентов, в некоторых случаях показывая производительность в десять раз превышающую производительность своих конкурентов.*

**- PC Magazine,** 26 января 1993 г.

*"Компьютер 486DX2-66 фирмы Zeos International, благодаря наилучшему сочетанию скорости, высококачественных компонентов и защиты от морального устаревания, буквально "растолкал" всех своих конкурентов..., представляет собой наиболее мощную, с возможностями дальнейшего роста машину по крайне конкурентоспособным ценам.*

**- Windows Magazine,** Май, 1993.

PC WORLD BEST BUY

*"Это быстрый, прекрасно спроектированный персональный компьютер с возможностью дальнейшего безграничного увеличения мощи. В результате – награда Best Buy ("Лучшая Покупка).*

**- PC World,** Февраль 1993 г., Best Buy

*"Блестящий дизайн материнской платы, великолепная производительность и тщательно продуманные мелочи... Эти качества выделяют ZEOS среди всей толпы".*

**- PC Computing,** Март 1993, Best Value ("Лучшая Ценность")

*"Преимущества в технической разработке и качестве... явно выделяют продукцию Zeos среди компьютеров его конкурентов, имеющих схожие цены".*

**- Forbes** (самый авторитетный журнал деловых кругов США), Март 1, 1993

Все поставки в СНГ осуществляются только с заводов фирмы Zeos International, Ltd. (США). Срок поставки стандартных конфигураций компьютеров – 7 дней. Все машины проходят 72–часовое "горячее" тестирование. С каждым компьютером Zeos Вы приобретаете лицензионные копии MS-DOS 6.0 и Windows 3.1 фирмы Microsoft.

**Гарантия на все компьютеры – 1 год.**

По всем вопросам закупок компьютеров фирмы Zeos International, Ltd. (настольные и серверы на основе процессоров фирмы Intel, начиная с 486SX25 и кончая 486DX2-66; блокнотного формата (цветные и моно VGA) на основе процессоров 386SL и 486SL; palmtop) обращаться в **Международный компьютерный клуб (МКК)** по адресу:
101813 Россия, Москва, Центр, Проезд Серова, 4 (Политехнический музей, подъезд 8, этаж 2, комн. 53, 54).

**Тел. (095) 921–09–02; (095) 925–46–67.**

# Ваша программа, на Вашем компьютере, но гораздо быстрее!

*Работу с этими и другими программами ускоряют сопроцессоры Intel*

## *Математические сопроцессоры фирмы Intel:*

— **привлекают крупнейших разработчиков программного обеспечения.** Свыше 600 таких компаний во всем мире ориентируются на математические сопроцессоры фирмы Intel, и уже свыше 2100 прикладных программ активно их используют;

— **предлагают проверенное решение для любого компьютера от XT до i486 SX.** Математические сопроцессоры фирмы Intel являются индустриальным стандартом. Во всем мире буквально миллионы пользователей с удовлетворением пожинают плоды повышенной производительности;

— **обеспечивают совместимость.** Поскольку сопроцессоры фирмы Intel разработаны специально для работы совместно с микропроцессорми Intel, на основе которых созданы все персональные компьютеры, они обеспечивают 100%—ную совместимость;

— **означают качество фирмы Intel.** Математические сопроцессоры, изготовленные фирмой Intel, являются наивысшим стандартом в мире и имеют пятилетнюю гарантию;

— **легко устанавливаются.** Надо лишь вставить сопроцессор в отведенное для него гнездо — и готово!

intel®